UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA DE TELECOMUNICACIONES

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA PREDECIR LAS PRECIPITACIONES FLUVIALES EN LA CIUDAD DE QUITO UTILIZANDO ELALGORITMO RANDOM FOREST

Trabajo de titulación previo a la obtención del Título de Ingeniero en Telecomunicaciones

AUTORES: ANDRÉS SEBÁSTIAN ARCINIEGA ZAVALA

JOEL FABIAN PASTAZ MONTES

TUTOR: LUIS GERMÁN OÑATE CADENA

Quito-Ecuador

2023

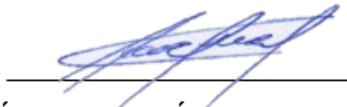
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Andrés Sebastian Arciniega Zavala documento de identificación N° 1719554543 y Joel Fabian Pastaz Montes con documento de identificación N° 1723029508; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro La Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 04 de agosto del año 2023.

Atentamente,



ANDRÉS SEBASTIÁN ARCINIEGA ZAVALA
1719554543

JOEL FABIÁN PASTAZ MONTES
1723029508

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros; Andrés Sebastián Arciniega Zavala, titular del documento de identidad 1719554543, y Joel Fabián Pastaz Montes, titular del documento de identidad 1723029508, certificamos en forma conjunta que el trabajo titulado, “Desarrollo de una aplicación móvil para predecir las precipitaciones fluviales en la ciudad de Quito utilizando el algoritmo Random Forest", ha sido escrito entera y exclusivamente por ambos autores. Al manifestar nuestra intención, otorgamos a la Universidad Politécnica Salesiana la titularidad de los derechos patrimoniales para que pueda hacer pleno uso de los derechos anteriormente otorgados. Como autores nos reservamos los derechos morales sobre el proyecto realizado de conformidad con lo establecido en la Ley de Propiedad Intelectual.

Quito, 04 de agosto del año 2023.

Atentamente,

ANDRÉS SEBASTIÁN ARCINIEGA ZAVALA
1719554543

JOEL FABIÁN PASTAZ MONTES
1723029508

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

El trabajo de título "DESAROLLO DE UNA APLICACIÓN MÓVIL PARA PREDECIR LAS PRECIPITACIONES PLUVIALES EN LA CIUDAD DE QUITO UTILIZANDO EL ALGORITMO RANFOM FOREST" fue elaborado bajo mi dirección, yo, Luis Germán Oñate Cadena, portador del documento de identidad 1712157401, docente de la Universidad Politécnica Salesiana, aquí declaro. El trabajo de grado bajo la opción de Proyecto Técnico que fue realizado por Andrés Sebastián Arciniega Zavala, portador de cédula de identidad 1719554543, y Joel Fabián Pastaz Montes**,** portador de cédula de identidad 1723029508, cumplió con todos los requisitos establecidos por la Universidad Politécnica Salesiana.

Quito, 0 4 de agosto del año 2023

Atentamente:

Ing. Luis Germán Oñate Cadena, MSc
171215740

DEDICATORIA

A nuestros padres, quienes nos han brindado su apoyo incondicional siendo nuestra mayor inspiración. Cada acción que tomamos requirió su apoyo, amor, compromiso y fe en nosotros. Su apoyo permanente al proyecto evidencia el éxito de sus esfuerzos. A nuestros profesores y consejeros, quienes a lo largo de la carrera académica nos compartieron sus conocimientos y experiencias. Agradecemos su compromiso, tolerancia y dirección en la estructuración de este proyecto, así como el amor por aprender y enseñar que nos han inculcado.

Andrés Sebastián Arciniega Zavala

Dedicamos este proyecto a todos aquellos que luchan por sus metas y encuentran la fuerza para superar los desafíos a través de su esfuerzo y perseverancia. Que este trabajo sirva como evidencia de que las metas sepueden lograr con trabajo duro.

Joel Fabian Pastaz Montes

ÍNDICE GENERAL

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN 2
CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN 3
DEDICATORIA 6
RESUMEN 1
ABSTRACT 8
INTRODUCCIÓN 9
CAPÍTULO 1 11
1.1. Planteamiento del Problema 11
1.2. Justificación 12
1.3. Objetivos 13
1.3.1. Objetivo General 13
1.3.2. Objetivos Específicos 13
1.4. Metodología 14
CAPÍTULO 2 15
2.1. Random Forest 15
2.2. Métodos de Ensamblaje 17
2.3. Bagging y Boosting 17
2.4. Entrenamiento Random Forest 18
2.5. Evaluación Confiabilidad Random Forest 18
2.6. Weatherstack 20
2.7. Postman 20
2.8. Expo 21
2.9. Framework React Native 21
2.10. API Key 21
CAPÍTULO 3 22
3.1. Diagramas de Flujo 22
3.1.1. Diagramas de Flujo de Home 22
3.1.2. Diagramas de Flujo de componente Pantalla 23
3.1.3. Diagramas de Flujo de componente Team 24
3.2. Resultados Obtenidos 25
3.2.1. La Exactitud 25

3.2.2. La Precisión 26
3.2.3. La Sensibilidad 26
3.2.4. La Especificidad 27
3.2.5. La tasa de falsos negativos 27
3.2.6. Matriz de Confusión del conjunto de prueba del 70% 28
3.2.7. La Exactitud 31
3.2.8. La Precisión 31
3.2.9. La Sensibilidad 32
3.2.10. La Sensibilidad 32
3.2.11. La tasa de falsos negativos 33
3.2.12. Matriz de Confusión del conjunto de prueba del 30% 34
3.3. Diagrama de Pastel: Test vs Real 36
3.3.1. La Sensibilidad 37
3.3.2. Error en la predicción 38
3.4. Fotografía de Weatherapp en la pantalla principal 38
3.4.1. Fotografía de la pestaña Home 39
3.4.2. Fotografía de la pestaña Proyecto 39
3.4.3. Fotografía de la pestaña Equipo 40
CAPÍTULO 4 41
4.1. Costo de Diseño 41
4.2. Costo de mano de obra 41
4.3. Costos adicional 42
4.4. Pregunta No 1 43
4.5. Pregunta No 2 43
4.6. Pregunta No 3 44
4.7. Pregunta No 4 45
4.8. Estimación de Ingresos 45
CAPÍTULO 5 47
5.1. Conclusiones 47
5.2. Recomendaciones 48
5.3. Bibliografía 49
5.3.1. Bibliografía Figuras 50
ANEXOS 51

RESUMEN

Quito tiene problemas con las precipitaciones pluviales, estas precipitaciones pueden resultar en inundaciones, deslizamientos de tierras, y otros desastres naturales que ponen en peligro vidas y causan grandes daños a la propiedad. La Ciudad no puede responder a estos eventos y prepararse para ellos de forma eficiente debido a la falta de un sistema eficiente de pronóstico de precipitaciones. Los enfoques tradicionales para el pronóstico del tiempo son con frecuencia insuficientemente precisos o no toman en cuenta las características únicas de Quito. Se sugiere que para abordar este problema, se cree una aplicación móvil que pronostique la precipitación de lluvia en la ciudad utilizando el algoritmo Random Forest. El algoritmo se refiere a un método de aprendizaje automático que relaciona diferentes modelos de predicción para aumentar la precisión de las predicciones. Esta aplicación recopilará información meteorológica en tiempo real, incluida la temperatura, la humedad, la presión del aire y más, e incorporarlo como datos para el algoritmo. La aplicación podrá producir pronósticos de precipitación pluviales con mayor precisión y previsión, ayudaría a los residentes de Quito y brindarles información más precisa sobre las posibles lluvias.

ABSTRACT

Like numerous other urban areas, Quito has issues with rainfall. This precipitation may result in landslides, floods, and other natural disasters that put lives in danger and significantly harm property. The City is unable to respond to these events and prepare for them in a sufficient manner due to the lack of an efficient river precipitation forecasting system. Conventional weather forecasting techniques frequently fall short in terms of accuracy or are not tailored to Quito's unique characteristics. Recommended to solve this problem, a mobile application should be created that employs the Random Forest algorithm to forecast river precipitation in the city. A machine learning method called the Random Forest algorithm combines several predictive models to increase the precision of predictions. The temperature, humidity, air pressure, and other real-time weather information would be gathered by this mobile app. and incorporate it into the Random Forest algorithm as input. Fluvial precipitation forecasts could be produced by the application with more accuracy and foresight. The mobile application would improve planning and decision-making for emergency management, preventive evacuations, and other disaster mitigation measures by giving Quito citizens and local authorities more precise information on potential river rainfall.

INTRODUCCIÓN

En particular, en áreas urbanas densamente pobladas como la ciudad de Quito, la precipitación pluvial es un fenómeno natural que puede tener un impacto significativo en las ciudades. Estas precipitaciones tienen el potencial de causar inundaciones repentinas, deslizamientos de tierra y otros eventos catastróficos que ponen vidas en peligro y provocan daños materiales significativos.

En esta situación, disponer de un sistema fiable de previsión de la precipitación de lluvia que es fundamental tanto para planificar la respuesta necesaria ante posibles desastres naturales como para tomar medidas preventivas. La precisión y adaptabilidad de las técnicas convencionales de pronóstico del tiempo a las características únicas de Quito son buenas, pero no siempre son correctas.

Para abordar este problema, es necesario utilizar nuevos métodos y herramientas para un mejor pronóstico de la precipitación. En este sentido, el motivo del actual diseño es crear una adaptación móvil que utilice el algoritmo Random Forest para pronosticar las lluvias en Quito.

La aplicación móvil propuesta facilitará la recopilación y análisis de datos meteorológicos, para pronosticar precipitaciones pluviales de un día para otro. Esta aplicación permite al ciudadano de Quito tener una herramienta útil para tomar decisiones informadas y poner en marcha medidas preventivas en caso de mal tiempo.

El documento consta de cinco capítulos que son los siguientes:

Capítulo 1 describe la declaración del problema, la justificación, el objetivo general, los objetivos específicos y la metodología.

Capítulo 2 describe los fundamentos teóricos, la información sobre el algoritmo Random Forest, así como los elementos empleados en la creación de la aplicación.

Capítulo 3 describe el algoritmo diseñado y las pruebas realizadas.

Capítulo 4 describe el análisis de los costos totales, se revela con el fin de evaluar la viabilidad del dispositivo implementado.

Capítulo 5 se encuentran las conclusiones, recomendaciones y bibliografía.

En el anexo 1 se encuentra el código de programación para predecir si habrá lluvia utilizando el lenguaje Python

CAPÍTULO 1

En este capítulo se expone la presentación del problema, la razón de ser del proyecto, los objetivos generales y específicos, así como la metodología empleada

1.1. Planteamiento del Problema

Si la tendencia del cambio climático continúa acelerándose, pudiendo tener un impacto negativo en la calidad de vida de las personas debido al aumento de las precipitaciones. Investigadores han informado recientemente que, a medida que la atmósfera se calienta cada vez más, retendrá una mayor cantidad de agua, lo que resultará en lluvias intensas a lo largo de un siglo. (Smith, 2017)

Dichas precipitaciones provocaron, problemas a nivel de tráfico, deslaves, inundaciones, desbordamientos de ríos. La ciudad de Quito se ve afectada por la congestión vehicular cada vez que hay lluvia sin contar con los accidentes que ha sucedo a causa de las lluvias excesivas. Lo que impulso la creación de tecnologías basadas en inteligencia artificial (IA).

La proyección de los niveles de agua futuros basada en la recopilación de información es realizada por la Inteligencia Artificial, la cual proporciona datos sobre las precipitaciones y el nivel de agua, junto con pronósticos para las próximas horas. Además, se destaca que esta tecnología puede optimizarse rápidamente incluso ante cambios en el entorno o la introducción de nueva infraestructura, requiriendo en tal caso un nuevo entrenamiento con datos de lluvia y nivel de agua

actualizados. Asimismo, se menciona que esta tecnología ofrece una precisión "equivalente o superior" a los métodos estándar utilizados para predecir los niveles de agua. (PIXABAY, 2019)

Con el fin de brindar información precisa para la toma de decisiones, se está desarrollando una aplicación basada en el algoritmo Random Forest. Este proyecto busca proporcionar una alternativa a las personas interesadas en la predicción de precipitaciones fluviales a través de algoritmos de inteligencia artificial.

1.2. Justificación

La adopción de decisiones en base a algoritmos de inteligencia artificial se ha incrementado en esta década. Dentro de las variables del clima se encuentran la temperatura, las precipitaciones de lluvia, y los vientos que afectan a la ciudad de Quito. Sin embargo, las lluvias son un factor crítico porque causan deslaves, inundaciones que afectan el tráfico.

Este método de predicción es innovador porque utiliza un algoritmo de inteligencia artificial denominado Random Forest para predecir en base a los históricos de las precipitaciones de la lluvia, la probabilidad de que llueva en Quito.

Otro aspecto innovador es que se crea una aplicación móvil que ayude a predecir si lloverá en Quito, ya que así podremos disminuir los accidentes y estar preparados de una manera más eficiente ante cualquier circunstancia.

La aplicación servirá como el gran aporte en la adopción de un sistema de alerta por impacto de desastres naturales, el cual es de gran importancia en la reducción de pérdidas patrimoniales y desastres ocasionados por lluvias intensas concentradas en toda la ciudad de Quito.

1.3. Objetivos

1.3.1. Objetivo General

Desplegar una aplicación móvil para predecir la probabilidad de lluvia en Quito en función de los históricos de temperatura y la precipitación pluvial.

1.3.2. Objetivos Específicos

Identificar las restricciones en el diseño de una aplicación móvil capaz de predecir el porcentaje de probabilidad de lluvia y analizar los algoritmos que podrían ser empleados para tal fin.

Crear una aplicación móvil basada en Random Forest para predecir la probabilidad de lluvia en Quito.

Comprobar la precisión del pronóstico contrastándolo con datos de otras aplicaciones o fuentes.

Establecer los costos de desarrollo de la aplicación para poder evaluar la viabilidad financiera de la aplicación.

1.4. Metodología

En el proyecto actual, se emplearán varias técnicas junto con la recuperación de datos meteorológicos precisos. A continuación, se establecerá la característica principal de Python de la aplicación utilizando esta metodología., demostrando cómo se relacionan Random Forest y la programación orientada a objetos. También se examinarán los costos asociados con la implementación de la aplicación, junto con un análisis del modelo Random Forest que se utilizará. Finalmente, a través de la experimentación, se confirmará el rendimiento de la aplicación al contrastarla con otras aplicaciones confiables como Google.

CAPÍTULO 2

Este capítulo describe los fundamentos teóricos, la información sobre el algoritmo Random Forest, así como los elementos empleados en la creación de la aplicación.

2.1. Random Forest

Considera un esquema de bosque aleatorio de conocimiento automático el cual utiliza árboles de decisión de manera individual cada uno entrenado en muestras no tan distintas a los datos de inicio, el pronóstico de un nuevo resultado se establece relacionando los pronósticos o estimaciones de todos los árboles de manera individual.

Los métodos de árbol constituyen en uno de los principales elementos de referencia en la industria de los pronósticos ya que estos brindan resultados precisos para una amplia gama de problemas.

Ventajas

- Los esquemas y/o modelos de bosque aleatorio permiten seleccionar automáticamente características relevantes.

- Estos modelos son aplicables tanto para problemas de regresión como en clasificación.

- Dado que son métodos no paramétricos, no se requiere que los datos sigan una distribución de manera específica.
- Son modelos especialmente útiles en los servicios de información ya que facilitan identificar de manera precisa y efectiva las variables críticas para realizar predicciones más precisas.

Desventajas

- Una limitación de los modelos de bosque aleatorio es que cuando se utilizan varios árboles, se pierde gran parte de la interpretación obtenida con los modelos de un solo árbol.

- Cuando se trabaja con predictores continuos, existe el riesgo de que se pierda información en la categorización en el momento de la división del nodo.

- Los modelos de bosque aleatorio se limitan a extrapolar dentro del rango de predictores observados en los datos de entrenamiento sin exceder estos límites.

Los modelos de bosques aleatorios se representan en la utilización de múltiples árboles de decisión, los cuales son construidos de forma individual a partir de muestras aleatorias extraídas de los datos de entrenamiento originales mediante el método de arranque. Cada árbol es entrenado con datos ligeramente distintos. Durante la construcción de cada árbol, las observaciones se distribuyen a través de las ramas hasta llegar a los nodos finales.

2.2. Métodos de Ensamblaje

Los métodos para equilibrar el sesgo y la varianza en el aprendizaje estadístico y los modelos de aprendizaje automático consisten en combinar múltiples modelos en uno para aumentar la precisión y la generalización de las predicciones.

2.3. Bagging y Boosting

En el Bagging, se entrena un conjunto de modelos utilizando subconjuntos diferentes de los datos de entrenamiento. Estos modelos se emplean para promediar las predicciones individuales y así obtener una predicción final.

En el Boosting, los modelos se presentan de manera secuencial y cada uno de ellos se busca corregir los errores de modelos anteriores.

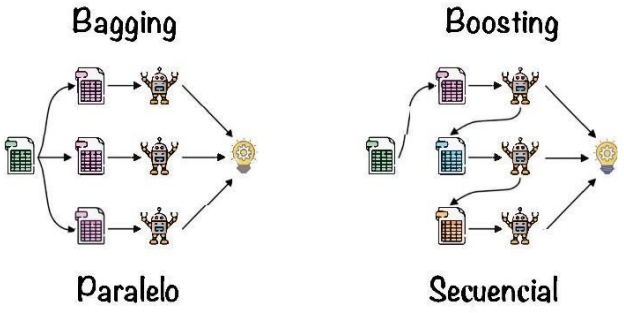


Figura.1 Bagging y Boosting
Fuente: (Ajitesh Kumar, 2022, p. 1)

En la figura 1 se observa una ilustración del funcionamiento de Bagging que funciona entrenando múltiples modelos en diferentes subconjuntos de datos y luego combinando las predicciones de esos modelos y Bossting entrenando una serie de modelos débiles y luego combinando las predicciones de esos modelos.

2.4. Entrenamiento Random Forest

Random Forest es una técnica basada en lotes que tiene como objetivo mejorar aún más la correlación entre los árboles generados. Para este propósito, antes de cada división del árbol, Random Forest contiene una selección aleatoria de un subconjunto de predictores.

Esto permite que otros predictores sean considerados en las divisiones y ayuda a correlacionar aún más los árboles generados., Random Forest evita este problema seleccionando aleatoriamente m vaticinio antes de estimar cada división.

Por lo tanto, la media de las divisiones (p−m) /p no considera el predictor influyente. Por lo tanto, se pueden optar por otros predictores. Juntar este paso adicional decora los árboles aún más para que su suma resulte una superior disminución de la varianza.

2.5. Evaluación Confiabilidad Random Forest

Se pueden usar varias métricas y métodos para evaluar la confiabilidad de un modelo de bosque aleatorio. Aquí hay algunas formas comunes de estimar modelos de bosques aleatorios:

Precisión o Exactitud: Esta es una métrica fundamental desarrollada para problemas de clasificación para entrenar o valorar el rentabilidad de un algoritmo de aprendizaje automático.

La matriz de confusión proporciona información detallada sobre la rentabilidad de un modelo de categorización, mediante la evaluación de los resultados de las predicciones. Esta matriz muestra la relación entre las categorías reales y las clases previstas por el método utilizado.

Figura 2. Matriz de confusión
Fuente: (Juan Ignacio Barrios Arce, 2019, p.1)

La Figura 2 muestra una representación gráfica de una matriz de confusión que presenta cuatro resultados posibles verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN).

2.6. Weatherstack

Aunque Weatherstack y Random Forest son ideas diferentes, se pueden usar juntas en escenarios de modelado predictivo o análisis de datos. Se puede acceder a la información meteorológica en vivo a través de la API del servicio de pronóstico de Weatherstack. Proporcione información meteorológica precisa y completa, incluida la temperatura, el vapor, la rapidez del viento, el pronóstico del tiempo a corto plazo, etc.

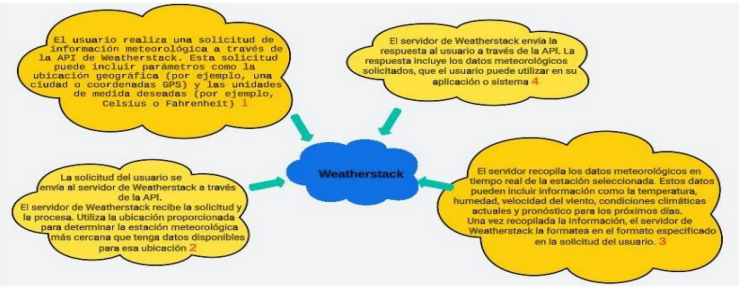


Figura.3 Weatherstack
Fuente: Joel Pastaz

En la figura 3 se observa un mapa conceptual sobre el proceso de Weatherstack para proporcionar información meteorológica precisa.

2.7. Postman

Es una organización que habilita y simplifica la creación y consumo de APIs. Esta herramienta es muy útil para la programación ya que permite probar y verificar el correcto funcionamiento de proyectos realizados por desarrolladores web. Proporciona la aptitud de gestionar la etapa de vida de la API.

2.8. Expo

Es una organización para el incremento rápido de aplicaciones React Native. Proporciona una capa sobre las API de React Native para que sean más fáciles de usar y administrar. También proporciona herramientas para facilitar la ejecución y prueba de aplicaciones React Native. Finalmente, se proporcionan servicios que generalmente solo están disponibles al instalar React Native y componentes de interfaz de usuario de terceros.

2.9. Framework React Native

Es una plataforma en la cual se utiliza código abierto el cual es diseñado por Meta Platforms el cual es dedicado para la construcción o diseño de aplicaciones lo que permite a los desarrolladores usar React con funcionalidad nativa en estas plataformas.

2.10. API Key

Una clave API es una cadena de caracteres única y secreta que se utiliza para autenticar y otorgar acceso a la API. Las claves de API son proporcionadas por proveedores de servicios y se utilizan para identificar y rastrear solicitudes de API realizadas por aplicaciones o usuarios.

Nuestra API Access Key: e38060d5c76240b9f0ce212460e84cbb

50.000 llamadas al mes

clima en tiempo real

CAPÍTULO 3

Este capítulo cubrirá la explicación de diagramas de flujo, matriz de confusión, resultados y el diseño de la aplicación.

3.1. Diagramas de Flujo

3.1.1. Diagramas de Flujo de Home

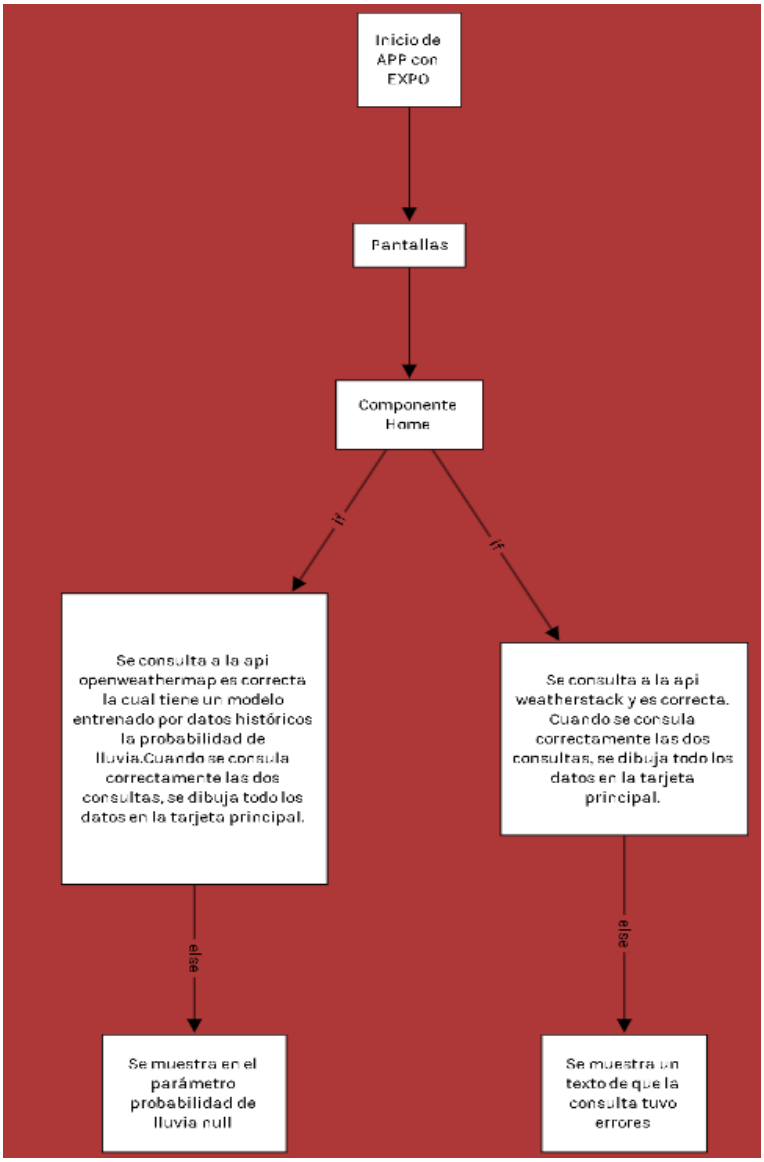


Fig.4 Diagrama de flujo componente Home.
Fuente: Andrés Arciniega

La Figura 4 muestra cómo el componente Inicio realza una serie de solicitudes a dos API diferentes: Weatherstack y openweathermap. Estas consultas se utilizan para obtener información sobre el clima y la probabilidad de lluvia. Si la solicitud de la API de Weatherstack es exitosa, todos los datos obtenidos se mostrarán en el mapa principal. Sin embargo, si la solicitud tiene un error, se muestra un texto que indica que la solicitud tiene errores resaltados en amarillo.

En el caso de una solicitud exitosa a la API de OpenWeatherMap, la probabilidad de lluvia en el mapa principal se determina mediante un modelo entrenado en datos históricos. Todos los datos, incluida la probabilidad de lluvia, se presentan en el mapa principal. En situaciones en las que no se puede determinar la probabilidad de lluvia, el parámetro correspondiente se muestra como nulo y se resalta en color amarillo.

3.1.2. Diagramas de Flujo de componente Descripción

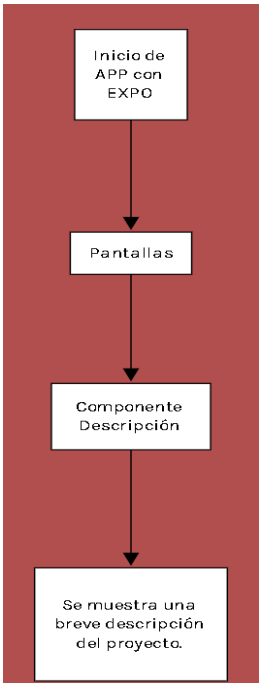


Fig.5. Diagrama de flujo componente Descripción
Fuente: Andrés Arciniega

En la Figura 5, en el componente descripción, se presenta un resumen conciso del proyecto. En esta sección se proporciona información sobre el propósito, objetivo y características destacadas del proyecto. La descripción ayuda a presentar a los usuarios y les da una idea de qué esperar de la aplicación.

3.1.3. Diagramas de Flujo de componente Team

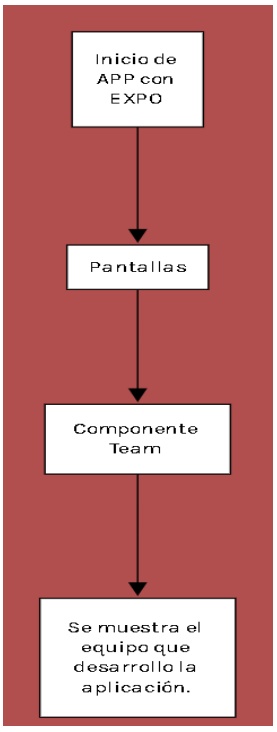


Fig.6 Diagrama de flujo componente Team
Fuente: Andrés Arciniega

En la Figura 6, dentro del componente equipo, se presenta la información acerca de los integrantes responsables del desarrollo de la aplicación. Esta sección generalmente incluye los nombres o alias de los miembros del equipo, junto con sus roles, responsabilidades y cualquier información pertinente sobre su experiencia o contribuciones al proyecto.

3.2. Resultados Obtenidos

En esta apartado, se analizan los resultados del modelado de desarrollo de aplicaciones y el enfoque matemático utilizado y se describen en detalle los principales procesos.

A continuación se presentan las tasas de precisión, veracidad, sensibilidad, especificidad y falsos negativos derivadas de la matriz de confusión de datos de prueba del 70 %.

3.2.1. La Exactitud

La exactitud se basa a la cantidad de pronósticos positivas que fueron correctas en relación con la cantidad total de predicciones realizadas.

$$\frac{(VP+VN)}{(VP+FP+FN+VN)}$$ ecuación (1)

$$\frac{(7886+5820)}{(7886+2342+1835+5820)} = 0.77$$

El valor obtenido de la Ecuación 1 es 0,77, lo que significa que el 77% de las predicciones positivas fueron correctas en relación con el número total de pronósticos realizados.

3.2.2. La Precisión

Se expresa mediante la relación entre los verdaderos positivos y todos los resultados positivos.

$$\frac{VP}{(VP+FP)}$$ ecuación (2)

$$\frac{7886}{(7886+2342)} = 0.77$$

El valor obtenido de la Ecuación 2 es 0,77, lo que significa que el 77% de las predicciones positivas fueron correctas en relación con el número total de predicciones realizadas.

3.2.3. La Sensibilidad

Se presenta como la proporción de verdaderos positivos relativa a la suma de verdaderos positivos y falsos negativos.

$$\frac{VP}{(VP+FN)}$$ ecuación(3)

$$\frac{7886}{(7886+1835)} = 0.81$$

El valor obtenido de la ecuación 8 es 0.81, lo cual indica que el 81% de los casos positivos se han identificado correctamente como verdaderos positivos.

3.2.4. La Especificidad

Se refiere a casos negativos correctamente clasificados por el algoritmo.

$$\frac{VN}{(VN+FP)} \qquad \text{ecuación(4)}$$

$$\frac{5820}{(5820 + 2342)} = 0.71$$

El resultado de la ecuación 4, es 0.71, significa que el 71% de los casos negativos se han clasificado correctamente como negativos en relación con el total de negativos.

3.2.5. La tasa de falsos negativos

La tasa de error, también conocida como la probabilidad de perder una prueba verdaderamente positiva, es un aspecto importante para considerar.

$$\frac{FN}{(FN+VP)} \qquad \text{ecuación (5)}$$

$$\frac{1835}{(1835 + 7886)} = 0.19$$

El valor derivado de la Ecuación 5 en presencia de falsos negativos es 0,19, lo que significa que hay un 19 % de probabilidad de que la prueba o el modelo no tengan un verdadero positivo.

3.2.6. Matriz de Confusión del conjunto de prueba del 70%

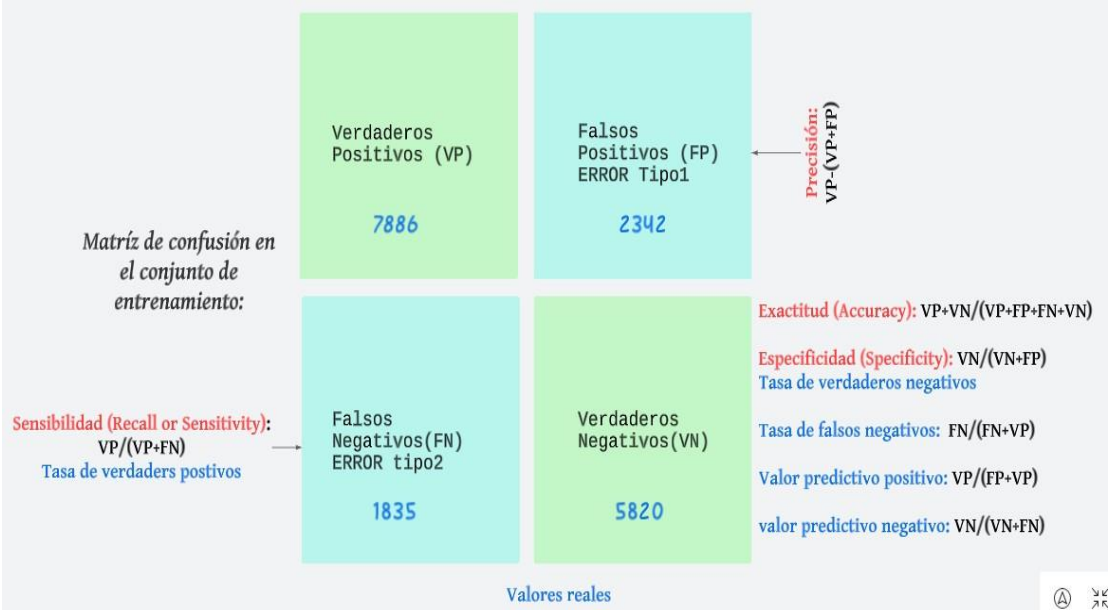


Fig.7 Matriz de confusión en el conjunto de entrenamiento al 70%
Fuente: Joel Pastaz

La figura 7 es una matriz de confusión, una tabla que muestra el rendimiento de un modelo de clasificación para predecir las clases reales de un grupo de datos, donde la matriz tiene NxN dimensiones. En este caso, la Matriz de Confusión se detalla en el grupo de prueba, que representa el 70% de los datos y proporciona información sobre los valores de Verdaderos Positivos (VP), Falsos Positivos (FP) y Falsos Negativos (FN).) y verdaderos negativos (TN).

3.2.6.1 Matriz de Confusión de Entrenamiento

Matriz	Clases	Precisión	Recall	F1-Score	Support
[[7886 2342]	0	0.81	0.77	0.79	10,228
[1835 5820]]	1	0.71	0.76	0.74	7,655
	accuracy	0.77	0.77	0.77	17,883
	macro avg	0.76	0.77	0.76	17,883
	weighted avg	0.77	0.77	0.77	17,883

Tabla.1 Matriz de confusión en el conjunto de entrenamiento al 70%

Fuente: Joel Pastaz

La Tabla 1 muestra la matriz de confusión en el grupo de adiestramiento, que representa el 70% de los datos. Esta matriz se utiliza para analizar la clasificación y evaluar el rendimiento del algoritmo de aprendizaje automático. Se crea comparando el pronóstico del modelo con los valores reales del grupo de datos. La tabla consta de seis columnas: matriz, clases, precisión, recuperación, puntaje F1 y soporte. Estas métricas proporcionan una medida de la calidad y el rendimiento.

3.2.6.2 Columna Matriz y Clases

La columna Clases hace referencia a las categorías o etiquetas asignadas a cada instancia. En el grupo de datos de preparación, cada instancia se etiqueta con una de estas clases y se recopila información sobre las capacidades de las instancias etiquetadas para el aprendizaje durante el entrenamiento del modelo. En este caso, la clase 0 representa una determinada categoría y la clase 1 pertenece a otra categoría específica.

3.2.6.3 Columna Precisión

La proporción de ejemplos positivos predichos correctamente se estima en comparación con todos los ejemplos realmente positivos. La precisión para la clase 0 es 0,81 y para la clase 1 es 0,71.

3.2.6.4 Columna Recall

En esta situación, se calcula la relación de ejemplos positivos predichos adecuadamente en coherencia con todos los ejemplos positivos verdaderos. La memoria para la clase 0 es 0,77 y para la clase 1 es 0,76.

3.2.6.5 Columna F1(f1-score)

Este es un indicador que relaciona precisión asegurando así un equilibrio entre los dos indicadores.

En esta situación, obtiene una precisión promedio de 0,76, una recuperación promedio de 0,77 y una puntuación F1 promedio de 0,76.

A continuación se incluye información sobre los parámetros de exactitud, precisión, sensibilidad, especificidad y tasa de falsos negativos basada la matriz de confusión contiene datos de prueba que representan el 30 % del total.

3.2.7. La Exactitud

La exactitud se refiere a la proporción de predicciones positivas que resultaron ser correctas.

$$\frac{(VP+VN)}{(VP+FP+FN+VN)} \qquad \text{ecuación (6)}$$

$$\frac{(3378+2438)}{(3378+1042+807+2438)} = 0.77$$

El valor derivado de la Ecuación 1 es 0,77, lo que significa que el 77% de las predicciones positivas fueron correctas en paralelo con el número total de pronósticos realizadas.

3.2.8. La Precisión

Se expresa mediante la relación entre los verdaderos positivos y todos los resultados positivos.

$$\frac{VP}{(VP+FP)} \qquad \text{ecuación(7)}$$

$$\frac{3378}{(3378 + 1042)} = 0.76$$

El valor derivado de la Ecuación 2 es 0,76, lo que significa que el 76% de los resultados clasificados como positivos en comparación con el número total de resultados positivos son verdaderos positivos.

3.2.9. La Sensibilidad

La sensibilidad se refiere a la equilibrio de resultados positivos verdaderos, mientras tanto que la especificidad se define como la equilibrio de resultados negativos verdaderos.

$$\frac{VP}{(VP+FN)}$$ ecuación (8)

$$\frac{3378}{(3378 + 807)} = 0.81$$

El resultado de la ecuación 8, es 0.81, significa que el 81% de los verdaderos positivos se han identificado correctamente en relación con los casos positivos.

3.2.10. La Sensibilidad

Se refiere a casos negativos clasificados correctamente por el algoritmo.

$$\frac{VN}{(VN+FP)}$$ ecuación (9)

$$\frac{2438}{(2438+1042)} = 0.70$$

El resultado de la ecuación 9, es 0.70, significa que el 70% de los casos negativos se han clasificado correctamente como negativos en relación con el total de negativos.

3.2.11. La tasa de falsos negativos

La tasa de error, también conocida como la probabilidad de perder una prueba verdaderamente positiva, es un aspecto importante para considerar.

$$\frac{FN}{(FN+VP)}$$ ecuación (10)

$$\frac{807}{(807+3378)} = 0.19$$

El valor obtenido de la ecuación 10, en presencia de una tasa de falsos negativos, es 0.19, lo cual indica que existe una probabilidad del 19% de que la prueba o modelo pase por alto un verdadero positivo.

3.2.12. Matriz de Confusión del conjunto de prueba del 30%

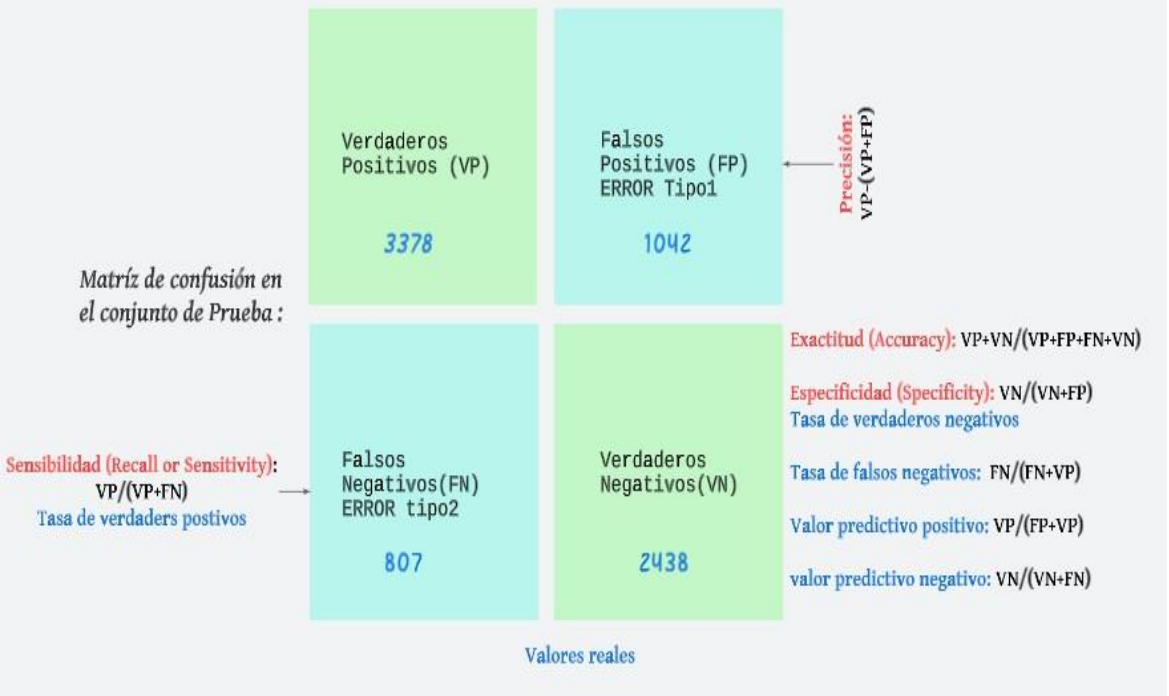


Fig.8 Matriz de confusión en el conjunto de prueba al 30%
Fuente: Joel Pastaz

En la Figura 8 hay una matriz de confusión, una tabla que muestra el rendimiento de un modelo de clasificación para predecir las clases reales de un grupo de datos. Esta matriz tiene dimensiones NxN, donde N representa el número de clases en el problema de clasificación. La figura muestra los detalles de la matriz de confusión en un grupo de pruebas que se ajusta al 30 % de los datos e incluye verdaderos positivos (VP), falsos positivos (FP), falsos negativos (FN) y resultados verdaderos, Negativo (V.N.).

3.2.12.1 Matriz de Confusión en el conjunto de prueba

Matriz	Clases	Precisión	Recall	F1-Score	Support
[[3378 1042]	0	0.81	0.76	0.79	4,420
[807 2438]]	1	0.7	0.75	0.73	3,245
	accuracy	0.76	0.76	0.76	7,665
	macro avg	0.75	0.76	0.76	7,665
	weighted avg	0.76	0.76	0.76	7,665

Tabla.2 Matriz de confusión en el conjunto de entrenamiento al 30%
Fuente: Joel Pastaz

La Tabla 2 muestra la matriz de confusión en el grupo de entrenamiento, que representa el 30% de los datos. Esta matriz se utiliza para el análisis de clasificación y para valorar el beneficio de un algoritmo de aprendizaje automático. Se crea comparando las predicciones del modelo con los valores reales del grupo de datos. La tabla consta de seis columnas: Matriz, Calificaciones, Precisión, Recordación, Puntaje F1 y Soporte. Estas métricas son útiles para evaluar la calidad y el rendimiento de un modelo.

3.2.12.2 Columna Matriz y Clases

La columna de clase se refiere a las categorías o etiquetas asignadas a cada instancia. Cada instancia en el grupo de datos de adiestramiento se etiqueta con una de estas clases y se recopila información sobre las características de las instancias etiquetadas que se aprenderán durante el entrenamiento del modelo. En este caso particular, la clase 0 pertenece a una categoría particular y la clase 1 a otra categoría particular.

3.2.12.3 Columna Precisión

La exactitud obtenida para la Clase 0 es de 0,81, lo que significa que el 81% de los ejemplos clasificados como Clase 0 pertenecen realmente a esa clase. Se registra una precisión de 0,70 para la Clase 1, lo que significa que el 70 % de los ejemplos asignados a la Clase 1 pertenecen realmente a esta clase.

3.2.12.4 Columna Recall

Para la clase 0, la respuesta es del 76 %, lo que significa que el 76 % de todos los ejemplos pertenecientes a la clase 0 se identificaron correctamente. Para la clase 1, la respuesta es del 75 %, lo que significa que el 75 % de todos los ejemplos de la clase 1 se identificaron correctamente.

3.2.12.5 Columna F1(f1-score)

La puntuación f1 es una medición combinada de exactitud y recuperación. Para clase 0 f1 es 79% y para clase 1 73%. En general, obtiene una exactitud promedio de 0,76, un recuerdo promedio de 0,77 y una puntuación F1 promedio de 0,76.

3.3. Diagrama de Pastel: Test vs Real

A continuación, se detalla la comparación de Prueba Vs Real con 500 datos de entrenamiento en un archivo csv en el cual se comparan 150 datos del Test son utilizados para comparar con el Real.

Fig.9 Porcentaje de predicciones de lluvia y no lluvia Test vs Real
Fuente: Joel Pastaz

La figura 9 muestra el porcentaje de precisión en la predicción vs error en la predicción en el cual se analizó las predicciones y los datos reales relacionados con la clase de lluvia.

A continuación, se describe los parámetros de precisión y error de predicción

3.3.1. La Sensibilidad

$$Precisión = \frac{(Número\ de\ predicciones\ correctas)}{(Número\ total\ de\ predicciones\ )} \qquad \text{ecuación (11)}$$

$$Precisión = \frac{118}{150} = 0.78.7$$

El resultado de la ecuación 11 es 0.78, significa que la precisión en la predicción de lluvia es aproximadamente 78.67%.

3.3.2. Error en la predicción

$$Error = 100 - (Precisión * 100) \qquad \text{ecuación (12)}$$

$$Error = 100 - (0.7866666666666666 * 100) = 21.33$$

El resultado de la ecuación 12 es 21.33 %, significa que el error en la predicción es aproximadamente 21.33%.

3.4. Fotografía de Weatherapp en la pantalla principal

Fig.10 WeatherApp
Fuente: Joel Pastaz

En la Figura 10, se observa icono de WeatherApp en la pantalla de inicio cuando lo abre, funciona sin errores perceptibles.

3.4.1. Fotografía de la pestaña Home

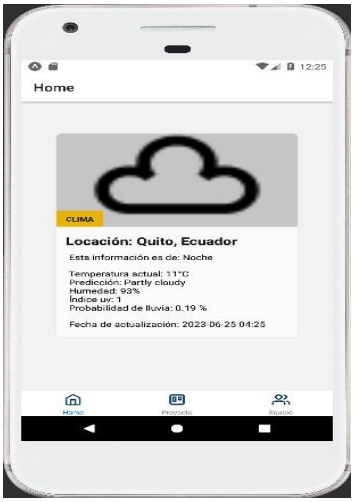


Fig.11 rain-probability-app
Fuente: Joel Pastaz

En la figura 11 se observa nuestra primera pestaña Home la cual muestra todos los datos como Temperatura, Predicción, Humedad etc.

3.4.2. Fotografía de la pestaña Proyecto

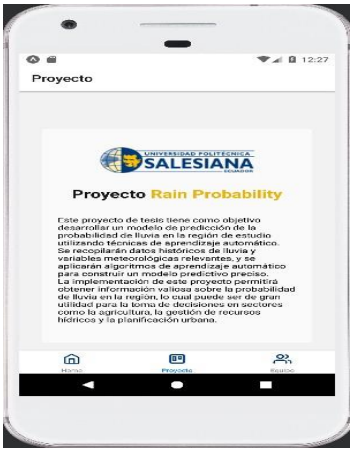


Fig.12 rain-probability-app
Fuente: Joel Pastaz

En la figura 12 se observa nuestra segunda pestaña Proyecto la cual nos da una breve explicación del desarrollo de la aplicación.

3.4.3. Fotografía de la pestaña Equipo

Fig.13 rain-probability-app
Fuente: Joel Pastaz

En la figura 13 se observa nuestra tercera pestaña Equipo la cual muestra el equipo de desarrolladores el cual realizo este proyecto.

CAPÍTULO 4

Este capítulo cubrirá el análisis de los costos totales, se revela con el fin de evaluar la viabilidad del dispositivo implementado.

A continuación se muestra una descripción de los parámetros relacionados con los costos de diseño, mano de obra y otros costos.

4.1. Costo de Diseño

	Cantidad	Costo
Costo por algoritmo	1	$1300
Total		$1300

Tabla.3 Costo de diseño
Fuente: Joel Pastaz

En tabla 3 se observa los costos de diseño de cada algoritmo individual que es de $1700, podemos interpretar que este costo representa el esfuerzo y los recursos dedicados al diseño de cada algoritmo específico.

4.2. Costo de mano de obra

	Costo por hora de trabajo	Total, de Horas	Costo Total
Costos de producción	$12	92H	$1.104
Total			$1.104

Tabla.4 Costo de mano de obra
Fuente: Joel Pastaz

En tabla 4 se observa el costo total asociado con las horas de trabajo dedicadas a la producción. En este caso, se menciona un "Costo por hora de trabajo" de $12 y un total de 92 horas de trabajo.

4.3. Costos adicional

Suscripción	Cantidad	Costo total
Weatherstack	1	$156
Google Play	1	$25
Total		$ 181

Tabla.5 Costo de diseño
Fuente: Joel Pastaz

En la tabla 5 se observa el costo adicional que se referirse Al gasto relacionado con la suscripción a Weatherstack y Google Play por un año.

A continuación, se presenta una descripción de la encuesta realizada a 100 personas con el objetivo de determinar cuántas de ellas tienen interés en nuestra aplicación, su disposición para pagar un precio y las características adicionales que les gustaría tener. Además, se recopila información sobre sus estimaciones de ingresos.

4.4. Pregunta No 1

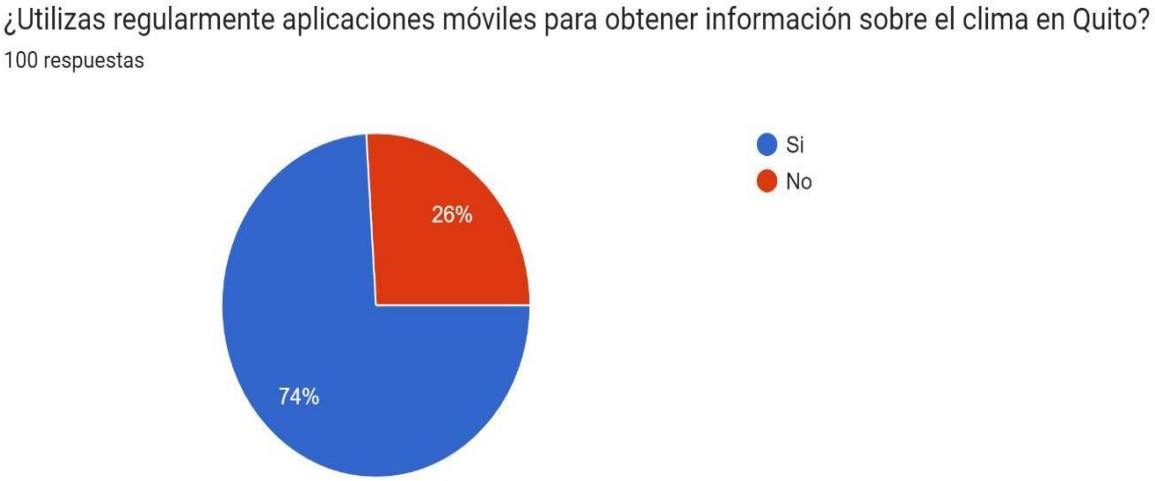


Fig.14 Diagrama de Pastel
Fuente: Joel Pastaz

La Figura 14 muestra que el 74% de los participantes respondieron positivamente. Este análisis muestra que la mayoría de los quiteños encuestados utilizan regularmente aplicaciones móviles para obtener información meteorológica.

4.5. Pregunta No 2

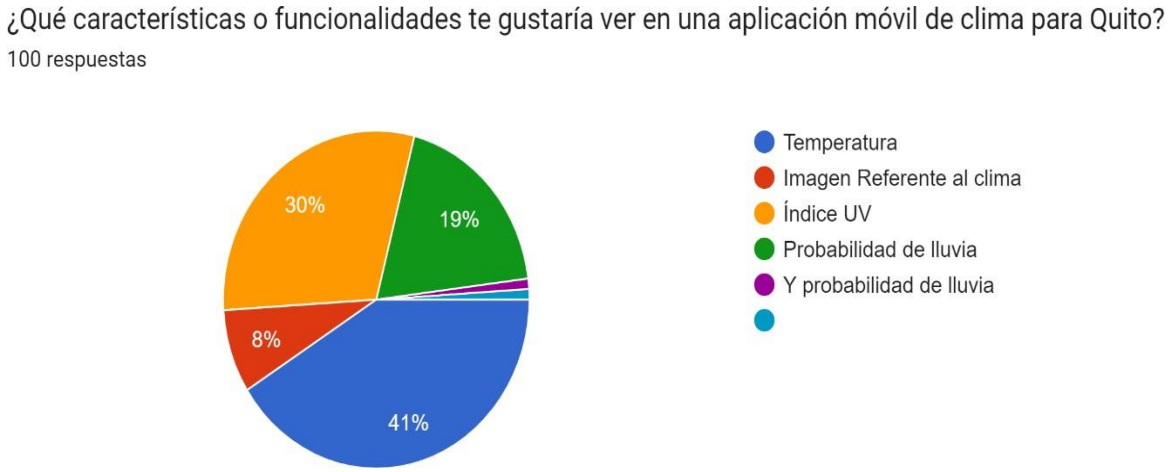


Fig.15 Diagrama de Pastel
Fuente: Joel Pastaz

En la figura 15 se observó que la temperatura es la característica más valorada por la mayoría de los encuestados en una aplicación móvil de clima para Quito, mientras que las imágenes, el índice UV y la probabilidad de lluvia son menos prioritarios.

4.6. Pregunta No 3

Fig.16 Diagrama de Pastel
Fuente: Joel Pastaz

En la Figura 16 puede ver que la mayoría de los posibles inversores están dispuestos a pagar por una aplicación de calidad que les proporcione información climática precisa. Sin embargo, también se identificó una minoría que prefiere opciones gratuitas o no siente la necesidad de pagar por este tipo de servicio.

4.7. Pregunta No 4

Fig.17 Diagrama de Pastel
Fuente: Joel Pastaz

La figura 17 indica que la mayor parte de los encuestados consideró que el precio de $3,50 por aplicación era justo y estaría dispuesto a pagarlo.

4.8. Estimación de Ingresos

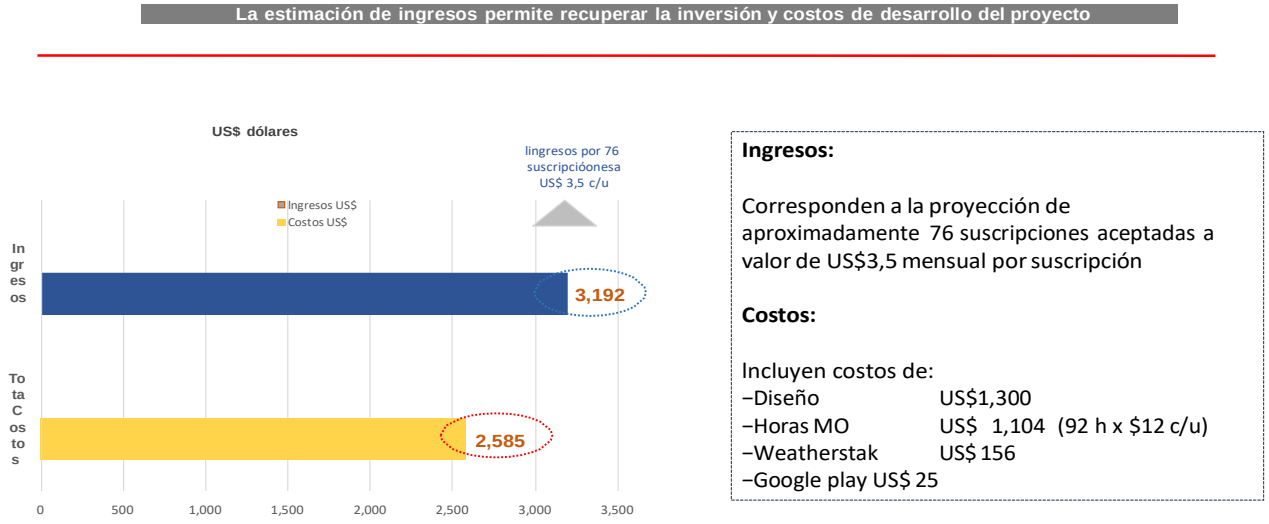


Fig.18 Diagrama de Barras
Fuente: Joel Pastaz

En la figura 18 se observó que los ingresos proyectados y los costos detallados estiman ingresos totales de $3,192 y costos totales de $2,585 en un año. Esto sugiere una ganancia neta estimada del 18% después de cubrir los costos asociados con la operación de la aplicación, lo que la posiciona como una aplicación aceptable en términos de rentabilidad.

CAPÍTULO 5

5.1. Conclusiones

La aplicación móvil desarrollada se presenta como una herramienta efectiva para predecir la precipitación fluvial en la ciudad de Quito. La utilización del algoritmo Random Forest ha demostrado ser apropiada para este propósito, brindando resultados precisos y confiables. Esta aplicación proporciona información en tiempo real sobre la precipitación, permitiendo a los usuarios estar preparados y tomar decisiones informadas en relación a medidas de prevención, seguridad y planificación.

Se realizaron las pruebas respectivas y se comprobó el correcto funcionamiento del dispositivo generando una precisión del 78.7% esto a la implementación de Ramdom Forest como inteligencia artificial por lo que se concluye que la implementación del dispositivo es aceptable.

La proyección de ingresos y los costos detallados indican que la aplicación tiene la capacidad de generar ingresos totales de $3,192 en un año, mientras que los costos totales se estiman en $2,585. Estas cifras sugieren una ganancia neta estimada después de cubrir los costos operativos de la aplicación, lo que la posiciona como una opción rentable desde una perspectiva financiera. La proyección de ingresos superiores a los costos indica que existe el potencial de obtener ganancias a través de la aplicación.

5.2. Recomendaciones

Es muy importante realizar un análisis completo de las características utilizadas como entrada para el algoritmo Random Forest. Es útil para investigar y experimentar con diversas variables meteorológicas como temperatura, humedad, presión barométrica, dirección y velocidad del viento, entre otras. Asimismo, se propone considerar la inclusión de características espaciales como la topografía y ubicación geográfica de la ciudad de Quito.

Importante realizar una validación exhaustiva y continua del modelo y la aplicación móvil. Compare pronósticos con datos reales y realice análisis de errores para identificar posibles oportunidades de mejora. Considere los comentarios de los usuarios y considere implementar un sistema de actualización y mejorar continuamente la aplicación.

5.3. Bibliografía

La época lluviosa aumenta el riesgo en estas parroquias de Quito. (2022, 15 octubre). el comercio. Recuperado 7 de abril de 2023, de https://www.elcomercio.com/actualidad/quito/riesgos-lluvias-quito-clima-desastres-siniestros.html

Random Forest python. (s. f.). https://www.cienciadedatos.net/documentos/py08_random_forest_python.html

Johanna Orellana Alvear - johanna.orellana@ucuenca.edu.ec. (s. f.-b). *Arboles de decisión y Random Forest.* https://bookdown.org/content/2031/ensambladores-random-forest-parte-i.html

Sklearn.ensemble.RandomForestClassifier. (s. f.-b). scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Koehrsen, W. (2019, 11 diciembre). *Random Forest in Python* - Towards Data Science. Medium. https://towardsdatascience.com/random-forest-in-python-24d0893d51c0?gi=78cedf9bd1ee

Sklearn.ensemble.RandomForestClassifier. (s. f.-a). scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

R, S. E. (2023, 24 marzo). *Understand Random Forest Algorithms With Examples* (Updated 2023). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

Machine Learning Random Forest Algorithm - Java point. (s. f.-c). www.javatpoint.com. https://www.javatpoint.com/machine-learning-random-forest-algorithm

S. (2023, 26 febrero). Random Forest Algorithm. Simplilearn.com.

https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorithm

Sandoval, C. (n.d.). *Inteligencia artificial, en el clima*. El Comercio. https://www.elcomercio.com/tendencias/tecnologia/inteligencia-artificial-clima-ciencia-tecnologia.html

5.3.1. Bibliografía Figuras

Kumar, A. (2022, noviembre 16*). Bagging vs Boosting Machine Learning methods. Data Analytics*. https://vitalflux.com/bagging-vs-boosting-machine-learning-methods/

Arce, J. I. B. (2019, julio 26*). La matriz de confusión y sus métricas. Juan Barrios.*

https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

ANEXOS

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os, pickle
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report



def dataPreprocess(file = 'seattleWeather_1948-2017.csv'):
  path = os.path.join(os.path.dirname(__file__), file)
  df = pd.read_csv(path)
  df = df.dropna()

  rainArr = []
  df['PRCP'] = df['PRCP'].astype('float')
  for i in df['RAIN']:
    #print(i,df['PRCP'][i])
    if i > 0:
      rainArr.append(np.random.randint(50,101))
    else:
      rainArr.append(np.random.randint(0,50))

  #print(rainArr, len(rainArr))
  df=df.drop('DATE',axis=1)
  rain = pd.Series(rainArr)
```

```
    df['RAIN'] = rain
    df = df.dropna()
    #print(df.head)
    y = df.pop('RAIN')
    X = df

    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30, random_state=42)

    return X_train, X_test, y_train, y_test

def trainer(X_train, X_test, y_train, y_test, model = RandomForestClassifier()):
    model.fit(X_train, y_train)
    print("modelo entrenado...")

    filename = "random_forest.pickle"

    # save model
    pickle.dump(model, open(filename, "wb"))
    print("Modelo guardado")
    #print("Matriz de confusión:\n",confusion_matrix(y_test, pred))
    #print("Reporte de clasificación:\n", classification_report(y_test,pred))
    #print("Importancia de las características:\n", model.feature_importances_)
    return model

def predictor(weatherVars, pop = -1):
    model = pickle.load(open("random_forest.pickle", "rb"))
    df = pd.DataFrame(weatherVars.reshape(1,-1), columns= ['PRCP', 'TMAX', 'TMIN'] )
    pred = model.predict(df)
    if pop >= 0:
```

```
    if  pop < 100:
      pred = pop + (pred**0)/100
    else:
      pred = pop - (pred**0)/100

  return pred

if __name___== '__main__':
  model = RandomForestClassifier()
  X_train, X_test, y_train, y_test = dataPreprocess()
  trainer(X_train, X_test, y_train, y_test, model)
  ----------------------------------------------------------------------------------------
from flet import *
import asyncio, pickle
import weatherMe as wt
import pandas as pd
import numpy as np

def main(page: Page):
   BG = '#041955'
   FWG = '#97B4FF'
   FG = '#3450A1'
   PINK = '#EB06FF'
   city = "Quito"
   page.window_width = 410 # window's width is 200 px
   page.window_height = 865 # window's height is 200 px
   page.window_resizable = True  # window is not resizable

   page.update()
```

```
#Load model and predict
icon = "assets/images/partialCloudy.png"
temp, mp, rainP = asyncio.run(wt.getweather(city))
#print(rainP, type(rainP))
categories = ['Rain Probability', 'Temperature', 'Moon Phase']
metrics = [f"{rainP}%", temp, mp]

circle = Stack(
   controls=[
      Container(
         width=100,
         height=100,
         border_radius=50,
         bgcolor='white12'
      ),
      Container(
         gradient=SweepGradient(
            center=alignment.center,
            start_angle=0.0,
            end_angle=3,
            stops = [0.5,0.5],
            colors = ['#00000000', PINK],
         ),
         width = 100,
         height=100,
         border_radius=50,
         content=Row(alignment='center',
                  controls=[
                     Image(width=90,
```

```
                        height=100,
                        border_radius=50,
                        #fit=ImageFit.CONTAIN,
                        src = f"assets/images/partialCloudy.png",

                        )
                ],)
        ),

    ]
)

def shrink(e):
    page_2.controls[0].width = 120
    page_2.controls[0].scale = transform.Scale(
                        0.8,
                        alignment = alignment.center_right
                    )
    page_2.controls[0].border_radius = border_radius.only(
        topLeft = 35,
        topRight = 0,
        bottomLeft = 35,
        bottomRight = 0
    )
    page_2.update()

def restore(e):
    page_2.controls[0].width = 400
    page_2.controls[0].border_radius = 35
```

```
    page_2.controls[0].scale = transform.Scale(
                                1,
                                alignment = alignment.center_right
                            )
    page_2.update()

update_location = Container(
    content = Container(
        on_click = lambda _: page.go('/'),
        height = 40,
        width = 40,
        content = Text('Actualizar'))
)

categories_card = Row(
    scroll = 'auto'
)



#Detalles
details = Column(
    height = 450,
    scroll = 'auto',
    controls = [
        Container(
            height = 320, width = 360,
            bgcolor = 'white12',
            border_radius = 20,
```

```
            content=Image(width=90,
                    height=90,
                    border_radius=20,
                    #fit=ImageFit.CONTAIN,
                    src = icon,)

        ),
        Container(height = 60, width = 360,
                bgcolor = 'white12',padding = padding.only(
                                    left = 50,
                                    top = 10),
                border_radius = 20,
                content = Text(metrics[0] + ' Rain Probability', size=20, weight=FontWeight.W_100))
    ]
)

page.update()

cat = ''
for i, category in enumerate(categories):
    if category != 'Moon Phase':
        cardType = Text(metrics[i], size=30, weight=FontWeight.W_100)
    else:
        cardType = Image(width=80,
                    height=80,
                    border_radius=20,
                    #fit=ImageFit.CONTAIN,
                    src = f"assets/images/{mp}.png",)
    categories_card.controls.append(
```

```
        Container(
            bgcolor = BG,
            height = 180,
            width = 200,
            padding = 15,
            border_radius = 20,
            content = Column(
                controls = [
                    Text(category),
                    Container(
                        width = 160,
                        height = 5,
                        bgcolor = 'white12',
                        border_radius = 20,
                        padding = padding.only(right = 30),
                        content = Container(
                            bgcolor = PINK,
                        ),
                    ),
                    cardType,
                ]
            )
        )
    )

first_page_contents = Container(
    content = Column(
        controls = [
            #icons bar
```

```
Row(
   alignment = 'spaceBetween',
   controls = [
      Container(
         on_click = lambda e: shrink(e),
         content = Icon(
            icons.MENU)),

      Row(
         controls = [
            Icon(icons.SEARCH),
            Icon(icons.NOTIFICATIONS_OUTLINED)
         ]
      )
   ]
),
#title page
Text(
   value = 'Rain Probability App'
),
#title segment 1
Text(
   value = 'CATEGORIES'
),
#Content segment 1
Container(
   padding = padding.only(top = 5),
   content = categories_card
),
```

```
            #space
            Container(height = 5),
            #Details of Categories
            Text("Details"),
            Stack(
               controls = [
                  details,
                  FloatingActionButton(top = 335, right = 10,
                     icon = icons.SEARCH,
                     on_click = lambda _: page.go('/location')
                  )
               ]
            )
         ]
      )
   )
page_1 = Container(
   width = 400,
   height = 855,
   bgcolor = BG,
   border_radius = 35,
   padding = padding.only(
               top = 60,
               left = 20,
               right = 200),
   content = Column(
      controls = [
         Row(alignment='end',
            controls=[
```

```
                Container(border_radius = 25,
                    padding = padding.only(top=13,left=13),
                    height = 50, width = 50,
                    border = border.all(color='white',width=2),
                    on_click = lambda e: restore(e),
                    content = Text('<')
                )
            ]
        ),
        Container(height=20),
        circle,
        Text(city, size=32, weight='bold'),
        Container(height=20),
        Row(controls=[
            Icon(icons.FAVORITE_BORDER_SHARP, color='white60'),
            Text('Rate us', size=15, weight=FontWeight.W_300, color='white', font_family='poppins')
        ])
    ]
  )
)
page_2 = Row(alignment = 'end',
  controls = [
    Container(
        width = 400,
        height = 850,
        bgcolor = FG,
        border_radius = 35,
        animate = animation.Animation(600, AnimationCurve.DECELERATE),
        animate_scale = animation.Animation(400, curve = 'decelerate'),
```

```
            padding = padding.only(
                top = 50,
                left = 20,
                right = 20,
                bottom = 5
            ),
            content = Column(
                controls = [
                    first_page_contents
                ]
            )
        )
    ]
)

container = Container(
    width = 400,
    height = 850,
    bgcolor = BG,
    border_radius = 35,
    content = Stack(
        controls = [
            page_1,
            page_2,
        ]
    )
)

pages = {
```

```
        '/': View(
                "/",
                [
                    container
                ],
            ),
        '/location':View(
                "/location",
                [
                    update_location
                ],
            ),
    }

    def route_change(route):
        page.views.clear()
        page.views.append(
            pages[page.route]
        )

    page.add(container)

    page.on_route_change = route_change
    page.go(page.route)


app(target=main, assets_dir='assets')
```

```
import python_weather
import asyncio
import os, requests, json
import pandas as pd
import numpy as np
import geopandas as gpd
import matplotlib.pyplot as plt
from geopy.geocoders import Nominatim
from geopy.exc import GeocoderTimedOut
from shapely.geometry import Point, Polygon
from sklearn.ensemble import RandomForestClassifier
import randForestRain as rft

async def getweather(city):
    # declare the client. format defaults to the metric system (celcius, km/h, etc.)
    async with python_weather.Client(format=python_weather.IMPERIAL) as client:

    # fetch a weather forecast from a city
        weather = await client.get(city)

    # returns the current day's forecast temperature (int)
    print(f"Temperatura: {(weather.current.temperature-32)*5/9} °C")
    moonP = []
    # get the weather forecast for a few days
    for forecast in weather.forecasts:
        print(forecast.date, forecast.astronomy)
        p = forecast.astronomy.moon_phase
        moonP.append(str(p))
```

```
        # hourly forecasts
        for hourly in forecast.hourly:
            print(f' --> {hourly!r}')


    #print(moonP)
    model = RandomForestClassifier()
    X_train, X_test, y_train, y_test = rft.dataPreprocess()
    rft.trainer(X_train, X_test, y_train, y_test, model)
    mint, maxt, prcp, pop = currentTime(city)
    varsW = np.array([[prcp, mint, maxt]])
    rainP = rft.predictor(varsW,pop)
    watherJs = probDaily(city)
    #icon = watherJs['current']['weather_icons']
    print(moonP[0])
    return "{:.2f} °C".format((weather.current.temperature-32)*5/9), moonP[0], rainP[0]#, icon


def plotMap(city):
    crs = {'init': 'epsg:4326'}
    try:
        geolocator = Nominatim(user_agent="weather", timeout=200)
        location = geolocator.geocode(city)
        if location:
            df = pd.DataFrame([location], columns=['City Name', 'Coordinates'])
            print(df.head())
            geometry = [Point(x[1], x[0]) for x in df['Coordinates']]
            geo_df = gpd.GeoDataFrame(df, crs=crs, geometry=geometry)
            print(geo_df.head())
```

```
    except GeocoderTimedOut as e:
        print("Error: geocode failed on input %s with message %s"%
            (city, e))

    countries_map = gpd.read_file('assets/maps/World_Countries/World_Countries.shp')
    f, ax = plt.subplots(figsize=(16, 16))
    countries_map.plot(ax=ax, alpha=0.4, color='grey')
    geo_df['geometry'].plot(ax=ax, markersize = 30,
                    color = 'r', marker = '^', alpha=.2)
    plt.show()

def probDaily(city):
    api_key = "b75bf6a1e5a842f674c7de02af02fa03"
    response = requests.get('http://api.weatherstack.com/current?access_key='+ api_key +'&query='+
city +'')

    data  =  response.json()
    rain_probability = data
    return rain_probability

def currentTime(city):
    url = "https://visual-crossing-weather.p.rapidapi.com/forecast"

    querystring =
{"aggregateHours":"24","location":city,"contentType":"json","unitGroup":"metric","shortColumnNames
":"0"}

    headers = {
        "X-RapidAPI-Key": "415f6009a0mshd00394d51de3b77p1927c7jsne0b1d5a255dd",
```

```
        "X-RapidAPI-Host": "visual-crossing-weather.p.rapidapi.com"
    }

    response = requests.get(url, headers=headers, params=querystring)

    data = response.json()['locations'][city]
    current = data['currentConditions']
    today = data['values'][0]
    #print(today)
    return today['mint'], today['maxt'], today['precip'], today['pop']

if __name__ == '__main__':
    city = 'Quito'
    #asyncio.run(getweather(city))
    #print(f"\nRain Probability OpenWeather:\n {probDaily(city)}")
    #plotMap(city)
```

```
import matplotlib.pyplot as plt
import numpy as np

# Datos de la matriz de confusión
train_confusion_matrix = np.array([[7886, 2342], [1835, 5820]])
test_confusion_matrix = np.array([[3378, 1042], [807, 2438]])

# Etiquetas de las clases
labels = ['0', '1']

# Crear la figura y los ejes
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 4))

# Gráfica de matriz de confusión para el conjunto de entrenamiento
train_cm = axes[0].imshow(train_confusion_matrix, interpolation='nearest', cmap=plt.cm.Blues)
axes[0].set_title('Matriz de confusión - Conjunto de entrenamiento')
axes[0].set_xticks(np.arange(len(labels)))
axes[0].set_yticks(np.arange(len(labels)))
axes[0].set_xticklabels(labels)
axes[0].set_yticklabels(labels)
axes[0].set_xlabel('Etiqueta predicha')
axes[0].set_ylabel('Etiqueta verdadera')

# Mostrar los valores en la matriz de confusión
```

```
for i in range(len(labels)):
    for j in range(len(labels)):
        axes[0].text(j, i, train_confusion_matrix[i, j], ha='center', va='center', color='red')

# Gráfica de matriz de confusión para el conjunto de prueba
test_cm = axes[1].imshow(test_confusion_matrix, interpolation='nearest', cmap=plt.cm.Blues)
axes[1].set_title('Matriz de confusión - Conjunto de prueba')
axes[1].set_xticks(np.arange(len(labels)))
axes[1].set_yticks(np.arange(len(labels)))
axes[1].set_xticklabels(labels)
axes[1].set_yticklabels(labels)
axes[1].set_xlabel('Etiqueta predicha')
axes[1].set_ylabel('Etiqueta verdadera')

# Mostrar los valores en la matriz de confusión
for i in range(len(labels)):
    for j in range(len(labels)):
        axes[1].text(j, i, test_confusion_matrix[i, j], ha='center', va='center', color='black')

# Agregar una barra de color a cada gráfica
fig.colorbar(train_cm, ax=axes[0])
fig.colorbar(test_cm, ax=axes[1])

# Ajustar los espacios entre subgráficas
plt.tight_layout()

# Mostrar la gráfica
plt.show()
```