

Flight Delay Prediction Using Machine Learning

Lagdhir Vaghela (18IT140)

Meet Vyas (18IT146)

Smt. Kundaben Dinsha Patel,
Department of Information Technology,
C.S.P.I.T.,
Changa, Anand, India.
18it140@charusat.edu.in
18it146@charusat.edu.in

Abstract

Flight delay is inevitable and it plays an important role in both profits and loss of the airlines. An accurate estimation of flight delay is critical for airlines because the results can be applied to increase customer satisfaction and incomes of airline agencies. Through machine learning, we can predict the delay of flight. It becomes very useful if we can predict the delay of flight. It can affect us in many ways. So in this project, we try to predict the flight delay.

Introduction

As air travel has a significant role in the economy of agencies and airports, it is necessary for them to increase the quality of their services. One of the important modern life challenges of airports and airline agencies is flight delay. In addition, delay in flight makes passengers concerned and this matter causes extra expenses for the agency and the airport itself. In 2007, the U.S. government had endured 31–40 billion dollar downsides due to flight delays. In 2017, 76% of the flights arrived on time. Where, in comparison to 2016, the percentage of on-time flights decreased by 8.5%. As some of the reasons for flight delays the following can be mentioned: security, weather conditions, shortage of parts and technical and airplane equipment issues and flight crew delays. Delay in flight is inevitable, which has too many negative economic effects on passengers, agencies and airports. Furthermore, delay can damage the environment through fuel consumption increment and also leads to emission of pollutant gases. In addition, the delay affects

the trade, because goods' transport is highly dependent on customer trust, which can increase or decrease the ticket sales, so that on-time flight leads to customer confidence. So that, flight prediction can cause a skillful decision and operation for agencies and airports, and also a good passenger information system can relatively satisfy the customer.

The Bureau of Transportation Statistics has recorded the nationwide flight operation data in the United States which provides valuable and reliable datasets for studying flight delay issues. Meanwhile, with the development of artificial intelligence, machine learning technology has been widely used in airport flight delay prediction. Machine learning technology involves multiple disciplines, such as probability, statistics, and computer science. Machine learning can break the limitations of mathematical formulas and improve the accuracy of flight delay prediction. In general, machine learning technology can be roughly divided into supervised learning, unsupervised learning, deep learning, reinforcement learning, and ensemble learning. Each of these learning methods has its characteristics.

Requirements

Recommended Operating Systems

- **Windows:** 7 or newer
- **MAC:** OS X v10.7 or higher
- **Linux:** Ubuntu

Hardware Requirements

- **Processor:** Minimum 1 GHz;
Recommended 2GHz or more
- **Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)**
- **Hard Drive:** Minimum 32 GB;
Recommended 64 GB or more
- **Memory (RAM):** Minimum 1 GB;
Recommended 4 GB or above

Implementation

We implement it using Google colab because, Google Colab is a Free tool or environment that facilitates interactive programming and data analysis using Python and other programming languages. And it's a web-based, making it an ideal solution for collaborating online.

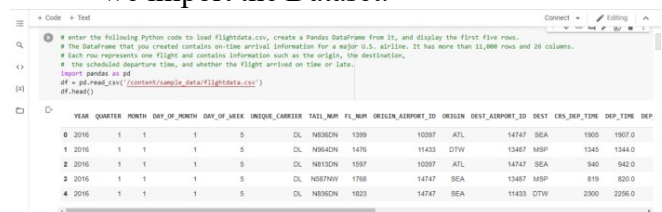
First, we learn about all the libraries that we used to develop this project.

- **Pandas:** pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- **math:** The Python Math Library provides us access to some common math functions and constants in Python, which we can use throughout our code for more complex mathematical computations. The library is a built-in Python module, therefore you don't have to do any installation to use it.
- **sklearn:** Scikit-learn(sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistency interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

- **matplotlib:** Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.
- **seaborn:** Seaborn is an open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily with data frames and the Pandas library. The graphs created can also be customized easily.
- **numpy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Create a Google Colab Notebook and Import Data

- First we create google colab notebook then we import the Dataset.



```
# Enter the following Python code to load flightdata.csv, create a Pandas DataFrame from it, and display the first five rows.
# The DataFrame that you created contains on-time arrival information for a major U.S. airline. It has more than 11,000 rows and 26 columns.
# Each row represents one flight and contains information such as the origin, the destination,
# the scheduled departure time, and whether the flight arrived on time or late.
import pandas as pd
df = pd.read_csv('content/sample_data/flightdata.csv')
df.head()
```

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	DEST_AIRPORT_ID	DEST	CRS_DEP_TIME	DEP_TIME	DEP
0	2016	1	1	1	5	DL	N638DN	1359	1028T	ATL	14747	SEA	1905	1907.0	
1	2016	1	1	1	5	DL	N964DN	1476	1143T	DTW	1348T	MSP	1345	1344.0	
2	2016	1	1	1	5	DL	N813DN	1087	1028T	ATL	14747	SEA	940	942.0	
3	2016	1	1	1	5	DL	N877NW	1768	1474T	SEA	1348T	MSP	819	820.0	
4	2016	1	1	1	5	DL	N638DN	1023	1474T	SEA	1143T	DTW	2300	2256.0	

- The DataFrame that we created contains on time arrival information for a major U.S. Airline. It has more than 11,000 rows and 26 columns. Each row represents one flight and contains information such as the origin, the destination, the scheduled departure time, and whether the flight arrived on time or late. We will look at the data more closely a bit later in this module.

Clean and prepare data

Before we prepare a dataset, we need to understand its content and structure. We have imported a dataset containing on-time arrival information for a major U.S. airline. That data included 26 columns and thousands of rows, with each row representing one flight and containing information such as the flight's origin, destination, and scheduled departure time. We also loaded the data into a notebook and used a simple Python script to create a Pandas DataFrame from it.

A Data Frame is a two-dimensional labeled data structure. The columns in a Data Frame can be of different types, just like columns in a spreadsheet or database table. It is the most commonly used object in Pandas.

One of the first things we want to know about a dataset is how many rows it contains. To get a count, type the following statement into an empty cell at the end of the notebook and run it:

```
[ ] # Confirm that the DataFrame contains 11,231 rows and 26 columns:  
df.shape  
(11231, 26)
```

Confirm that the Data Frame contains 11,231 rows and 26 columns.

Now take a moment to examine the 26 columns in the dataset. They contain important information such as the date that the flight took place (YEAR, MONTH, and DAY_OF_MONTH), the origin and destination (ORIGIN and DEST), the scheduled departure and arrival times (CRS_DEP_TIME and CRS_ARR_TIME), the difference between the scheduled arrival time and the actual arrival time in minutes (ARR_DELAY), and whether the flight was late by 15 minutes or more (ARR_DEL15).

Here is a complete list of the columns in the dataset. Times are expressed in 24-hour military time. For example, 1130 equals 11:30 a.m. and 1500 equals 3:00 p.m.

Column	Description
YEAR	Year that the flight took place
QUARTER	Quarter that the flight took place (1-4)
MONTH	Month that the flight took place (1-12)
DAY_OF_MONTH	Day of the month that the flight took place (1-31)
DAY_OF_WEEK	Day of the week that the flight took place (1=Monday, 2=Tuesday, etc.)
UNIQUE_CARRIER	Airline carrier code (e.g., DL)
TAIL_NUM	Aircraft tail number
FL_NUM	Flight number
ORIGIN_AIRPORT_ID	ID of the airport of origin
ORIGIN	Origin airport code (ATL, DFW, SEA, etc.)
DEST_AIRPORT_ID	ID of the destination airport
DEST	Destination airport code (ATL, DFW, SEA, etc.)
CRS_DEP_TIME	Scheduled departure time
DEP_TIME	Actual departure time
DEP_DELAY	Number of minutes departure was delayed
DEP_DEL15	0=Departure delayed less than 15 minutes, 1=Departure delayed 15 minutes or more
CRS_ARR_TIME	Scheduled arrival time
ARR_TIME	Actual arrival time
ARR_DELAY	Number of minutes flight arrived late
ARR_DEL15	0=Arrived less than 15 minutes late, 1=Arrived 15 minutes or more late
CANCELLED	0=Flight was not cancelled, 1=Flight was cancelled
DIVERTED	0=Flight was not diverted, 1=Flight was diverted
CRS_ELAPSED_TIME	Scheduled flight time in minutes
ACTUAL_ELAPSED_TIME	Actual flight time in minutes
DISTANCE	Distance traveled in miles

The dataset includes a roughly even distribution of dates throughout the year, which is important because a flight out of Minneapolis is less likely to be delayed due to winter storms in July than it is in January. But this dataset is far from being “clean” and ready to use. Let’s write some Pandas code to clean it up.

- One of the most important aspects of preparing a dataset for use in machine learning is selecting the “feature” columns that are relevant to the outcome you are trying to predict while filtering out columns that do not affect the outcome, could bias it in a negative way, or might produce multicollinearity. Another important task is to eliminate missing values, either by deleting the rows or columns containing them or replacing them with meaningful values.

```
[ ] # Confirm that the output is "True," which indicates that there is at least one missing value somewhere in the dataset.  
df.isnull().values.any()  
True
```

Confirm that the output is “True,” which indicates that there is at least one missing value somewhere in the dataset.

- The next step is to find out where the

missing values are. To do so, execute the following code:

```
[ ] # Number of missing values in each column
df.isnull().sum()

YEAR                0
QUARTER             0
MONTH               0
DAY_OF_MONTH        0
DAY_OF_WEEK         0
UNIQUE_CARRIER     0
TAIL_NUM            0
FL_NUM              0
ORIGIN_AIRPORT_ID   0
ORIGIN              0
DEST_AIRPORT_ID     0
DEST                0
CRS_DEP_TIME        0
DEP_TIME            107
DEP_DELAY            107
DEP_DEL15           107
CRS_ARR_TIME        0
ARR_TIME            115
ARR_DELAY            188
ARR_DEL15           188
CANCELLED           0
DIVERTED             0
CRS_ELAPSED_TIME    0
ACTUAL_ELAPSED_TIME 188
DISTANCE            0
Unnamed: 25         11231
dtype: int64
```

- Curiously, the 26th column (“Unnamed: 25”) contains 11,231 missing values, which equals the number of rows in the dataset. This column was mistakenly created because the CSV file that you imported contains a comma at the end of each line. To eliminate that column, add the following code to the notebook and execute it.

```
[ ] # The DataFrame with column 26 removed(Unnamed: 25)
df = df.drop('Unnamed: 25', axis=1)
df.isnull().sum()

YEAR                0
QUARTER             0
MONTH               0
DAY_OF_MONTH        0
DAY_OF_WEEK         0
UNIQUE_CARRIER     0
TAIL_NUM            0
FL_NUM              0
ORIGIN_AIRPORT_ID   0
ORIGIN              0
DEST_AIRPORT_ID     0
DEST                0
CRS_DEP_TIME        0
DEP_TIME            107
DEP_DELAY            107
DEP_DEL15           107
CRS_ARR_TIME        0
ARR_TIME            115
ARR_DELAY            188
ARR_DEL15           188
CANCELLED           0
DIVERTED             0
CRS_ELAPSED_TIME    0
ACTUAL_ELAPSED_TIME 188
DISTANCE            0
dtype: int64
```

- The DataFrame still contains a lot of missing values, but some of them aren’t useful because the columns containing them are not relevant to the model that we are building. The goal of that model is to predict whether a flight you are considering booking is likely to arrive on time. If we know that the flight is likely to be late, we might choose to book another flight.

The next step, therefore, is to filter the dataset to eliminate columns that aren’t relevant to a predictive model. Pandas

provide an easy way to filter out columns we don’t want. Execute the following code in a new cell at the end of the notebook:

```
[ ] # The output shows that the DataFrame now includes only the columns that are relevant to the model,
# and that the number of missing values is greatly reduced:
df = df[["MONTH", "DAY_OF_MONTH", "DAY_OF_WEEK", "ORIGIN", "DEST", "CRS_DEP_TIME", "ARR_DEL15"]]
df.isnull().sum()
# The filtered DataFrame

MONTH              0
DAY_OF_MONTH       0
DAY_OF_WEEK        0
ORIGIN             0
DEST              0
CRS_DEP_TIME       0
ARR_DEL15          188
dtype: int64
```

- The only column that now contains missing values is the ARR_DEL15 column, which uses 0s to identify flights that arrived on time and 1s for flights that didn’t. Use the following code to show the first five rows with missing values:

```
# The only column that now contains missing values is the ARR_DEL15 column,
# which uses 0s to identify flights that arrived on time and 1s for flights that didn't.

# Pandas represents missing values with NaN, which stands for Not a Number.
# The output shows that these rows are indeed missing values in the ARR_DEL15 column.
df[df.isnull().values.any(axis=1)].head()

MONTH  DAY_OF_MONTH  DAY_OF_WEEK  ORIGIN  DEST  CRS_DEP_TIME  ARR_DEL15
177    1             9            MSP  SEA       701         NaN
178    1            10            MSP  DTW      1348         NaN
184    1            10            MSP  DTW       625         NaN
210    1            10            DTW  MSP      1200         NaN
478    1            22            SEA  JFK      2305         NaN
```

Pandas represent missing values with NaN, which stands for Not a Number. The output shows that these rows are indeed missing values in the ARR_DEL15 column.

- The reason these rows are missing ARR_DEL15 values are that they all correspond to flights that were canceled or diverted. We could call dropna on the DataFrame to remove these rows. But since a flight that is canceled or diverted to another airport could be considered “late,” let’s use the fillna method to replace the missing values with 1s.

Use the following code to replace missing values in the ARR_DEL15 column with 1s and display rows 177 through 184:

```
[ ] # Let's use the fillna method to replace the missing values with 1s.
df = df.fillna({'ARR_DEL15': 1})
df.iloc[177:185]

MONTH  DAY_OF_MONTH  DAY_OF_WEEK  ORIGIN  DEST  CRS_DEP_TIME  ARR_DEL15
177    1             9            MSP  SEA       701         1.0
178    1            10            DTW  JFK      1527         0.0
179    1            10            MSP  DTW      1348         1.0
180    1            10            DTW  MSP      1540         0.0
181    1            10            JFK  ATL      1325         0.0
182    1            10            JFK  ATL       610         0.0
183    1            10            JFK  SEA      1615         0.0
184    1            10            MSP  DTW       625         1.0
```

The dataset is now “clean” in the sense that missing values have been replaced and the list of columns has been narrowed to those most relevant to the model. But you’re not finished yet. There is more to do to prepare the dataset for use in machine learning.

Now, we will “bin” the departure times in the CRS_DEP_TIME column and use Pandas’ get_dummies method to create indicator columns from the ORIGIN and DEST columns.

- Use the following command to display the first five rows of the Data Frame:

```
# Observe that the CRS_DEP_TIME column contains values from 0 to 2359 representing military times.
df.head()
```

	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_DEP_TIME	ARR_DEL15
0	1	1	5	ATL	SEA	1905	0.0
1	1	1	5	DTW	MSP	1345	0.0
2	1	1	5	ATL	SEA	940	0.0
3	1	1	5	SEA	MSP	819	0.0
4	1	1	5	SEA	DTW	2300	0.0

Confirm that the numbers in the CRS_DEP_TIME column now fall in the range 0 to 23.

- Now use the following statements to generate indicator columns from the ORIGIN and DEST columns, while dropping the ORIGIN and DEST columns themselves:

```
# Now use the following statements to generate indicator columns from the ORIGIN and DEST columns,
# while dropping the ORIGIN and DEST columns themselves:
df = pd.get_dummies(df, columns=['ORIGIN', 'DEST'])
df.head()
```

	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_DEP_TIME	ARR_DEL15	ORIGIN_ATL	ORIGIN_DTW	ORIGIN_MSP	ORIGIN_SEA	DEST_ATL	DEST_DTW	DEST_MSP	DEST_SEA
0	1	1	5	1905	0.0	1	0	0	0	0	0	0	0
1	1	1	5	1345	0.0	0	1	0	0	0	0	0	1
2	1	1	5	940	0.0	1	0	0	0	0	0	0	0
3	1	1	5	819	0.0	0	0	0	0	1	0	0	1
4	1	1	5	2300	0.0	0	0	0	0	1	0	1	0

The dataset looks very different than it did at the start, but it is now optimized for use in machine learning.

Build Machine Learning Model

To create a machine learning model, we need two datasets: one for training and one for testing. In practice, we often have only one dataset, so we split it into two. Now, we will perform an 80–20 split on the DataFrame we prepared in the task so we can use it to train a machine learning model. We will also separate the DataFrame into feature columns and label columns. The former contains the columns used as input to the model while the latter contains the column that the model will attempt to predict in this case, the ARR_DEL15 column, which indicates whether a flight will arrive on time.

- Again Open that Notebook.

- Use the following statements to bin the departure times:

```
[ ] # Use the following statements to bin the departure times:
# Confirm that the numbers in the CRS_DEP_TIME column now fall in the range 0 to 23:
import math

for index, row in df.iterrows():
    df.loc[index, 'CRS_DEP_TIME'] = math.floor(row['CRS_DEP_TIME'] / 100)
df.head()
```

	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_DEP_TIME	ARR_DEL15
0	1	1	5	ATL	SEA	19	0.0
1	1	1	5	DTW	MSP	13	0.0
2	1	1	5	ATL	SEA	9	0.0
3	1	1	5	SEA	MSP	8	0.0
4	1	1	5	SEA	DTW	23	0.0

Build Machine Learning Model

The first statement imports scikit-learn’s train_test_split helper function. The second line uses the function to split the DataFrame into a training set containing 80% of the original data, and a test set containing the remaining 20%. The random_state parameter seeds the random-number generator used to do the splitting, while the first and second parameters are DataFrames containing the feature columns and the label column.

```
[ ] from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(df.drop('ARR_DEL15', axis=1), df['ARR_DEL15'], test_size=0.2, random_state=1)
```

- In a new cell at the end of the notebook, enter and execute the following statements:

Build Machine Learning Model

The first statement imports scikit-learn’s train_test_split helper function. The second line uses the function to split the DataFrame into a training set containing 80% of the original data, and a test set containing the remaining 20%. The random_state parameter seeds the random-number generator used to do the splitting, while the first and second parameters are DataFrames containing the feature columns and the label column.

```
[ ] from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(df.drop('ARR_DEL15', axis=1), df['ARR_DEL15'], test_size=0.2, random_state=1)
```

The first statement imports scikit-learn’s train_test_split helper function. The second line uses the function to split the DataFrame into a training set containing 80% of the original data, and a test set containing the remaining 20%. The random_state parameter seeds the random number generator used to do the splitting, while the first and second parameters are DataFrames containing the feature columns and the label column.

- train_test_split returns four DataFrames. Use the following command to display the number of rows and columns in the DataFrame containing the feature columns used for training:

```
# display the number of rows and columns in the DataFrame containing the feature columns used for training:
train_x.shape
```

(8984, 14)

- Now use this command to display the number of rows and columns in the DataFrame containing the feature columns used for testing:

```
[ ] # display the number of rows and columns in the DataFrame containing the feature columns used for testing:
test_x.shape
```

(2247, 14)

One of the benefits of using scikit-learn is that you don’t have to build these models or implement the

algorithms that they use by hand. Scikit-learn include a variety of classes for implementing common machine learning models. One of them is `RandomForestClassifier`, which fits multiple decision trees to the data and uses averaging to boost the overall accuracy and limit over fitting.

- Execute the following code in a new cell to create a `RandomForestClassifier` object and train it by calling the `fit` method.

```
[ ] # Your model will be a binary classification model that predicts whether a flight will arrive on-time or late
# ("binary" because there are only two possible outputs).
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state=13)
model.fit(train_x, train_y)

RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                        criterion='gini', max_depth=None, max_features='auto',
                        max_leaf_nodes=None, max_samples=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=100,
                        n_jobs=None, oob_score=False, random_state=13, verbose=0,
                        warm_start=False)
```

The output shows the parameters used in the classifier, including `n_estimators`, which specifies the number of trees in each decision-tree forest, and `max_depth`, which specifies the maximum depth of the decision trees. The values shown are the defaults, but we can override any of them when creating the `RandomForestClassifier` object.

- Now call the `predict` method to test the model using the values in `test_x`, followed by the `score` method to determine the mean accuracy of the model:

```
[ ] # Now call the predict method to test the model using the values in test_x,
# followed by the score method to determine the mean accuracy of the model:
predicted = model.predict(test_x)
model.score(test_x, test_y)

0.8642634623943035
```

The mean accuracy is 86%, which seems good on the surface. However, mean accuracy isn't always a reliable indicator of the accuracy of a classification model. Let's dig a little deeper and determine how accurate the model really is that is, how adept it is at determining whether a flight will arrive on time.

There are several ways to measure the accuracy of a classification model. One of the best overall measures for a binary classification model is the Area under Receiver Operating Characteristic Curve (sometimes referred to as "ROC AUC"), which essentially quantifies how often the model will make a correct prediction regardless of the outcome.

Before you compute the ROC AUC, we must

generate prediction probabilities for the test set. These probabilities are estimates for each of the classes, or answers, the model can predict. For example, `[0.88199435, 0.11800565]` means that there's an 89% chance that a flight will arrive on time (`ARR_DEL15 = 0`) and a 12% chance that it won't (`ARR_DEL15 = 1`). The sum of the two probabilities adds up to 100%.

- Run the following code to generate a set of prediction probabilities from the test data:

```
[ ] # generate a set of prediction probabilities from the test data:
from sklearn.metrics import roc_auc_score
probabilities = model.predict_proba(test_x)
```

- Now use the following statement to

Generate a ROC AUC score from the probabilities using scikit-learn's `roc_auc_score` method:

```
[ ] # generate an ROC AUC score from the probabilities using scikit-learn's roc_auc_score method:
roc_auc_score(test_y, probabilities[:, 1])

0.7014819895830565
```

- Use the following code to produce a confusion matrix for your model:

```
# produce a confusion matrix for your model:
from sklearn.metrics import confusion_matrix
confusion_matrix(test_y, predicted)

array([[1903, 33],
       [ 272, 39]])
```

- Scikit-learn contain a handy method named `precision_score` for computing precision. To quantify the precision of your model, execute the following statements:

```
# quantify the precision of your model
from sklearn.metrics import precision_score

train_predictions = model.predict(train_x)
precision_score(train_y, train_predictions)

1.0
```

- Scikit-learn also contain a method named `recall_score` for computing recall. To measure our model's recall, execute the following statements:

```
# measure your model's recall, execute the following statements:
from sklearn.metrics import recall_score

recall_score(train_y, train_predictions)

0.9992012779552716
```

Visualize Output of Model

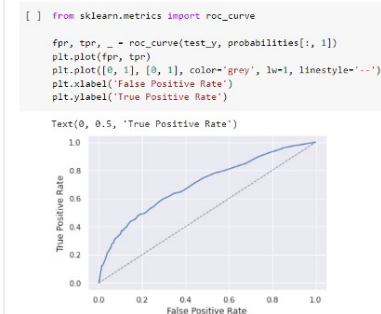
- Execute the following statements in a new cell at the end of the notebook. Ignore any warning messages that are displayed related to font caching:

Visualize Output of Model

```
[ ] In [ ]: %matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

The first statement is one of several magic commands supported by the Python kernel that you selected when you created the notebook. It enables Jupyter to render Matplotlib output in a notebook without making repeated calls to show. And it must appear before any references to Matplotlib itself. The final statement configures Seaborn to enhance the output from Matplotlib.

- To see Matplotlib at work, execute the following code in a new cell to plot the ROC curve for the machine-learning model we built in the previous task:



The dotted line in the middle of the graph represents a 50–50 chance of obtaining a correct answer. The blue curve represents the accuracy of our model. More importantly, the fact that this chart appears at all demonstrates that we can use Matplotlib in a notebook.

The reason we built a machine-learning model is to predict whether a flight will arrive on time or late. Now, we will write a Python function that calls the machine-learning model we built in the task to compute the likelihood that a flight will be on time. Then we will use the function to analyze several flights.

- Enter the following function definition in a new cell, and then run the cell.

```
def predict_delay(departure_date_time, origin, destination):
    """
    From datetime input, determine
    """
    try:
        departure_date_time_parsed = datetime.strptime(departure_date_time, '%d/%m/%Y %H:%M:%S')
        return "Error parsing datetime - {}".format(e)
    except ValueError as e:
        month = departure_date_time_parsed.month
        day = departure_date_time_parsed.day
        day_of_week = departure_date_time_parsed.isoweekday()
        hour = departure_date_time_parsed.hour

    origin = origin.upper()
    destination = destination.upper()

    input = [{"DEST": month,
              "DAY_OF_WEEK": day_of_week,
              "DAY_OF_WEEK_HOUR": hour,
              "ORIGIN_DEST": 1 if origin == "ATL" else 0,
              "DEST_ATL": 1 if origin == "ATL" else 0,
              "DEST_JFK": 1 if origin == "JFK" else 0,
              "DEST_HOU": 1 if origin == "HOU" else 0,
              "DEST_MIA": 1 if origin == "MIA" else 0,
              "DEST_MSP": 1 if origin == "MSP" else 0,
              "DEST_SFO": 1 if destination == "ATL" else 0,
              "DEST_ATL": 1 if destination == "JFK" else 0,
              "DEST_JFK": 1 if destination == "HOU" else 0,
              "DEST_HOU": 1 if destination == "MIA" else 0,
              "DEST_MIA": 1 if destination == "MSP" else 0,
              "DEST_SFO": 1 if destination == "SEA" else 0}]

    return model.predict_proba(input)[0][0]
```

This function takes as input a date and time, an origin airport code, and a destination airport code, and returns a value between 0.0 and 1.0 indicating the probability that the flight will arrive at its destination on time. It uses the machine-learning model we built in the previous task to compute the probability. And to call the model, it passes a DataFrame containing the input values to `predict_proba`. The structure of the DataFrame exactly matches the structure of the DataFrame we used earlier.

- Use the code below to compute the probability that a flight from New York to Atlanta on the evening of October 1 will arrive on time. The year we enter is irrelevant because it isn't used by the model.

```
[ ] In [ ]: predict_delay('1/10/2018 21:45:00', 'JFK', 'ATL')

0.88
```

- Modify the code to compute the probability that the same flight a day later will arrive on time:

```
[ ] In [ ]: predict_delay('2/10/2018 21:45:00', 'JFK', 'ATL')

0.87
```

- Now modify the code to compute the probability that a morning flight the same day from Atlanta to Seattle will arrive on time:

```
[ ] In [ ]: predict_delay('2/10/2018 10:00:00', 'ATL', 'SEA')

0.99
```

We now have an easy way to predict, with a single line of code, whether a flight is likely to be on time or late.

Execute the following code to plot the probability of on-time arrivals for an evening flight from JFK to ATL over a range of days:



Summary

In this module, we have learned how to:

- Create a notebook in Google Colab
- Import data into a notebook using Libraries
- Use Pandas to clean and prepare data
- Use scikit-learn to build a machine-learning model
- Use Matplotlib to visualize the results

Pandas, scikit-learn, and Matplotlib are among the most popular Python libraries on the planet. With them, we can prepare data for use in machine learning, build sophisticated machine-learning models from the data, and chart the output. Colab Notebooks provide a ready-made environment for using these libraries.

References

- <https://vaghelalucky12.medium.com/predict-flight-delays-by-creating-a-machine-learning-model-in-python-c36774577f19>
- <https://medium.com/analytics-vidhya/using-machine-learning-to-predict-flight-delays-e8a50b0bb64c>
- <https://youtu.be/2z8KarxxCwY>

Project Paper

ORIGINALITY REPORT

27%
SIMILARITY INDEX

25%
INTERNET SOURCES

21%
PUBLICATIONS

%
STUDENT PAPERS

PRIMARY SOURCES

1	<p>Maryam Farshchian Yazdi, Seyed Reza Kamel, Seyed Javad Mahdavi Chabok, Maryam Kheirabadi. "Flight delay prediction based on deep learning and Levenberg-Marquart algorithm", Journal of Big Data, 2020</p> <p>Publication</p>	7%
----------	--	-----------

2	<p>www.hindawi.com</p> <p>Internet Source</p>	3%
----------	--	-----------

3	<p>www.tutorialspoint.com</p> <p>Internet Source</p>	2%
----------	--	-----------

4	<p>mhcc.edu</p> <p>Internet Source</p>	2%
----------	--	-----------

5	<p>stackabuse.com</p> <p>Internet Source</p>	2%
----------	--	-----------

6	<p>journalofbigdata.springeropen.com</p> <p>Internet Source</p>	1%
----------	--	-----------

7	<p>www.ijert.org</p> <p>Internet Source</p>	1%
----------	--	-----------

8	<p>towardsdatascience.com</p> <p>Internet Source</p>	
----------	--	--

1 %

9

repozitorij.unizg.hr

Internet Source

1 %

10

tableau.toanhoang.com

Internet Source

1 %

11

docs.microsoft.com

Internet Source

1 %

12

"Beginning VB .NET 1.1 Databases", Springer
Science and Business Media LLC, 2005

Publication

1 %

13

Poornachandra Sarang. "Artificial Neural
Networks with TensorFlow 2", Springer
Science and Business Media LLC, 2021

Publication

1 %

14

www.coursemerit.com

Internet Source

1 %

15

Trevor Z. Dorn-Wallenstein, James R. A.
Davenport, Daniela Huppenkothen, Emily M.
Levesque. "Photometric Classifications of
Evolved Massive Stars: Preparing for the Era
of Webb and Roman with Machine Learning",
The Astrophysical Journal, 2021

Publication

<1 %

16

Vinita Silaparasetty. "Chapter 1 Getting
Started: Installation and Troubleshooting",

<1 %

17

"Business Intelligence", Springer Science and Business Media LLC, 2021

Publication

<1 %

18

Omar Alam, Anshuman Kush, Ali Emami, Parisa Pouladzadeh. "Predicting irregularities in arrival times for transit buses with recurrent neural networks using GPS coordinates and weather data", Journal of Ambient Intelligence and Humanized Computing, 2020

Publication

<1 %

19

Randy Betancourt, Sarah Chen. "Python for SAS Users", Springer Science and Business Media LLC, 2019

Publication

<1 %

20

hdl.handle.net

Internet Source

<1 %

21

www.overleaf.com

Internet Source

<1 %

22

"Advances in Computing and Network Communications", Springer Science and Business Media LLC, 2021

Publication

<1 %

23

"HCI International 2020 – Late Breaking Papers: Universal Access and Inclusive Design", Springer Science and Business Media LLC, 2020

Publication

<1 %

24

faq-courses.com

Internet Source

<1 %

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off