MODERN DATA VISUALISATION FOR AIR TRAFFIC MANAGEMENT

ABHIRAM SREEKANTHAM (19BCE2294)

OBJECTIVE

Air traffic at airports is affected by various factors. The capacity of an airport and the demand at a certain point in time are serious parameters that account for a big extent to aircraft delay and related variables. It has already been proven a lot of times that weather is another important impact in this regard. Although weather cannot be controlled by us humans, the knowledge of how weather affects the air traffic at an airport can be very helpful to optimize air traffic management. Data mining helps us to gain that knowledge. Usually, the very first step in data mining is data visualization

Considering a dataset of airline statistics, I employ Machine learning Algorithms to predict future possible flight delays so that travellers can plan their flights suitably, also the airports can modify their services accordingly so.

INTRODUCTION

- The airline industry is characterized by the complexity of its operations. So, when a delay occurs on a flight, the impact of it can become very important, even to create an effect snowball on all operational activities and resources involved. A delay can be caused by mechanical breakage, bad conditions weather, or security reasons. its effect is not negligible and must be absorbed as soon as possible in order to limit the consequences. The options, in order to restore the situation when a delay occurs, are very restricted. So, the first is to catch up by accelerating the affected flight and the second is to take no action and therefore, to assume all the costs that result.
- Since flying schedule is so important to any traveller, a flight delay may cause the traveller to reschedule or even worse, miss a transaction flight. For that cause, it's important to have an idea of how much time was wasted on flight delays, which airline have a reputation of the most delayed flight or which time of the year is better to avoid flying and find alternatives as a mean of transportation. All these questions will be answered according to a detailed analysis of the data of all US airlines arrival times, departure, reasons of delays etc. All this data is provided by the BTS (Bureau of Transportation Statistics) and after different and multiple researches about the projects problem, its discovered that flight delays is a serious issue where different articles and papers analyse it.
- Here, Machine Learning is used for the project. Machine Learning helps us identify the patterns in the data which might not be visible to our naked eye. It then is used to make useful predictions and correct

recommendations based on the data into consideration.

LITERATURE REVIEW

[1] In this paper, we have described two recent techniques that provide visualization of high-dimensional data. As a practical application, we have shown mappings of real high-dimensional weather and aviation data. The results which were obtained by using MDSpolar and density-based MDS clearly show the impact of the weather on arriving aircraft, but also impact of the traffic demand during extreme weather situations. Subject of future research should be a speedup of density-based MDS on the one hand and the consideration of different weather scenarios on the other hand. [2] Flight delay is a problem that occurs every day in US airports. In 2007, nearly one in four airline flights arrived at its destination over 15 minutes late (Ball et al, 1). A flight is considered delayed when it arrived 15 or more minutes than the schedule. The air congestion is increasing daily which reached more than 87,000 flight per day across the United states in one day (Miller,1).

Increase in travel delays vary from each airport to another in the US. Different causes like security, weather and even technical problems are the main reasons for these delays. This paper analyses a variety of delays causes for more than 30 airlines in the United States.

Along with this, an analysis of different types of raw data like departure delay time, arrival delay time, origin and even the number of cancellation due to weather, will answer different questions for travellers around the United States.

Finally, time series modelling for 3 airlines was done to predict delays. [3] Causes of flight delays and cancellations:

Living abroad for almost 15 years, I am a frequently seen guest at many airports. Unfortunately, I am regularly facing delayed or even cancelled flights.

Who doesn't?

The author is at the point that he is often surprised if his plane takes off on time and arrives on time. Luckily, international flights gather up some leverage so as to make up for lost time by pushing that gas pedal. The author believes many people don't realize that flight delays and cancellations put significant strain on the air transportation system and cost airlines, passengers and others millions of dollars each year.

The NEXTOR study which was performed by the University of California Berkeley's Institute of Transportation Studies found that flight delays and cancellations cost the US economy an estimated 33 billion dollar each year of which passengers bear ~ 15 billion.

One in four flights arrives or departs at least 15 minutes late. What is the actual cause for these unpopular delays?

The weather, a missed start window? Using publicly available data from the Research and Innovative Technology Administration Bureau of Transportation Services (RITA) (limited to August 2014), the overall goal of this particular project is to investigate general basis of flight's delay stats and identify airlines and airports with the highest delay rate.

Lastly, I analyzed the average time of delay for American Airlines. Based on my box plot, two different average delay time for passengers

were unusually long at two airports: HNL (~90 minutes) and XNA (~75 minutes). [4] Covid-19 pandemic has seriously impacted the world. In order to slow the pandemic's spread, countries issued travel restrictions, lockdows, social distancing mandates, etc., and therefore, a lot of businesses have been impacted. The traveling agency is one of the industries that has been hit very critically. Because of these travelling restrictions, the number of flights decreased all around the world dramatically. In this post, I will analyse and visualize the flight data.

So, the conclusion is that:

-Flight traffic decreased a lot at the start of the Covid-19 pandemic and has been recovered gradually.

-Low-cost airline (Easyjet) almost stop the flights from March to June, then recovered gradually since June.

-Cargo flights were never impacted.

[5] Nowadays, as there is a rapid increase in the amount of air traffic, so does the compulsion for virtuous, globally correlated and interoperable Air Traffic Management systems. We are searching for a well-regulated and widely harmonized Air Traffic Management framework, supported by a worthwhile and viable Communications, Navigation and Surveillance (CNS) infrastructure. In today's aeronautics practice, a human air traffic controller monitors and directs many aircraft flying through its baptized airspace zone. Here, Machine Learning is used for the project. Machine Learning helps us identify the patterns in the data which might never be visible to our naked eye. After that it is used to make useful predictions along with correct recommendations based on the data into consideration. Android provides us with a rich application-framework that allows us to build very innovative apps and games for mobile devices in a Java language environment. Here, Android is used to make a proper application of ATC.

PROBLEM STATEMENT

One in four flights arrives or departs at least 15 minutes late.

What is the actual cause for these unpopular delays?

The weather or a missed start window?

Using publicly available data from the Research and Innovative Technology Administration Bureau of Transportation Services (RITA), the overall goal of this project is to investigate the general basis of flight delays and identify airlines and airports with the highest delay rate.

- To determine those airports with the highest number of arriving flights per month.
- To define the proportion of delayed flights in the U.S. per month (no daily data available)
- To investigate the different causes of flight delays.
- To identify airlines and airports with the highest percentage of delayed flights

PROPOSED METHOD

The relationship between the various attributes of the flight statistics is analyzed through various plots like Bar charts and Stacked bar charts, Line charts and Pie charts.

First, we read and clean the data as usual, then to build the ML model we import scikit-learn's train_test_split helper function. Then we use the function to split the DataFrame into a training set containing x% of the original data, and a test set containing the

remaining (100-x) %. The random_state parameter seeds off 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.

create a RandomForestClassifier object and train it by calling the fit method.

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.

One of the best overall measures for a binary classification model is 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. Generate an ROC AUC score from the probabilities using scikit-learn's roc_auc_score method.

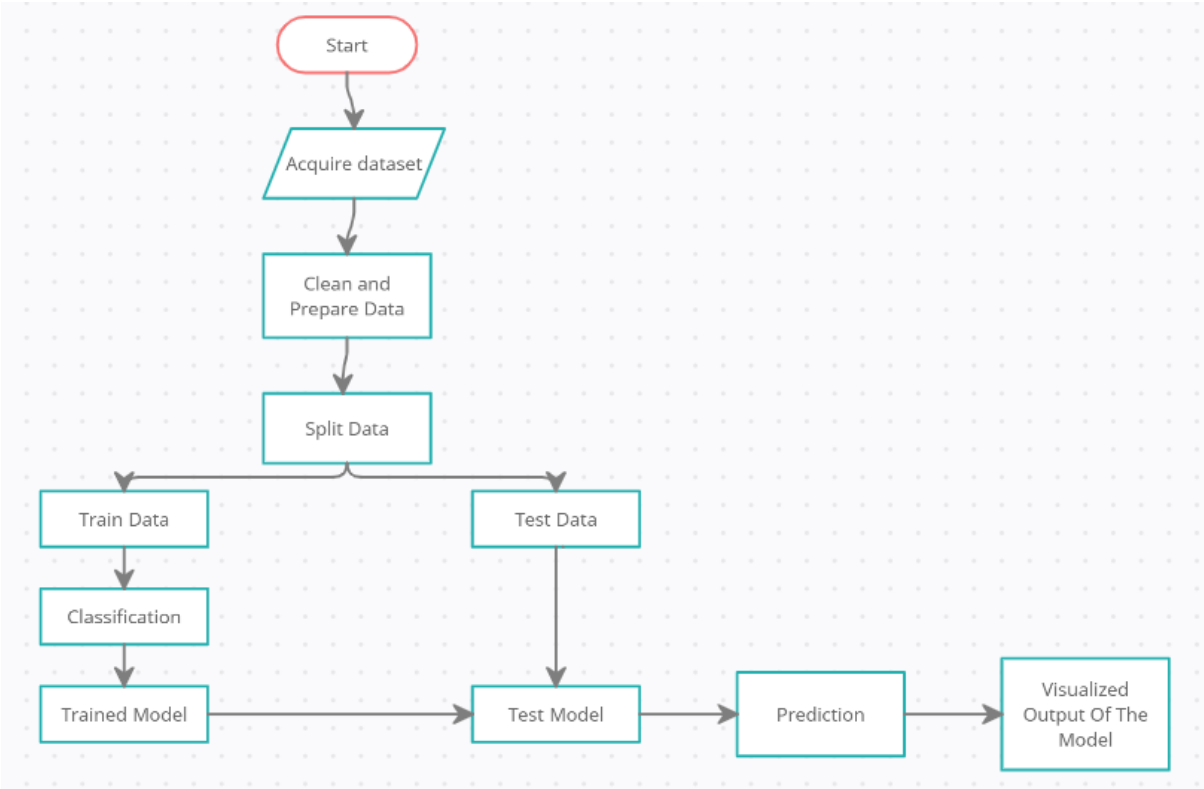
The output from the score method reflects how many of the items in the test set the model predicted correctly.

You can learn more about the model's behavior by generating a confusion matrix, also known as an error matrix. The confusion matrix quantifies the number of times each answer was classified correctly or incorrectly. Specifically, it quantifies the number of false positives, false negatives, true positives, and true negatives. Produce a confusion matrix for the model.

Other measures of accuracy for a classification model include precision and recall. Suppose the model was presented with three on-time arrivals and three delayed arrivals, and that it correctly predicted two of the on-time arrivals, but incorrectly predicted that two of the delayed arrivals would be on time. In this case, the precision would be 50% (two of the four flights it classified as being on time actually were on time), while its recall would be 67% (it correctly identified two of the three on-time arrivals).

Now, we can move on to defining our function and visualizing the output.

PROPOSED FLOW DIAGRAM



DATASET DESCRIPTION

The DataFrame that I took (flightdata.csv) 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. Data mining analysis was applied on the dataset and a subset of the actual dataset was done.

METHODOLOGY IMPLEMENTATION

First, we read and clean the data as usual, inspect any remove any null values, select the most important and concerning variables to the project and redefine the dataframe, replace all NaNs with 1 so that it'll be easier to deal with all numeric values while training the dataframe or defining the function, generate indicator columns from the ORIGIN and DEST columns, while dropping the ORIGIN and DEST columns themselves, now the dataframe is ready. Then we import scikit-learn's train_test_split helper function. Then we use the function to split the DataFrame into a training set containing x% of the original data, and a test set containing the remaining (100-x) %. The random_state parameter seeds off 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 train_x.shape() command to display the number of rows and columns in the DataFrame containing the feature columns used for training. Then use test_x.shape() command to display the number of rows and columns in the DataFrame containing the feature columns used for testing, if you called shape on the other two DataFrames,

train_y and test_y we would get only the row details of the DataFrame.

There are many types of machine learning models. One of the most common is the regression model, which uses one of a number of regression algorithms to produce a numeric value — for example, a person's age or the probability that a credit-card transaction is fraudulent.

We will train a classification model, which seeks to resolve a set of inputs into one of a set of known outputs. An example of a classification model is that one that examines e-mails and classifies them as "spam" or "not spam." Our model will be the binary classification model which predicts whether a flight will arrive on-time or late ("binary" because there are only two possible outputs).

One of the advantages of using scikit-learn is, you don't have to build these models — or implement the algorithms that they use — by hand. Scikit-learn includes 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 overfitting. We execute a code in a new cell to create a RandomForestClassifier object and train it by calling the fit method, 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 you can override any of them when creating the RandomForestClassifier object.

Next 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. 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.

There are several ways to measure the accuracy of a classification model. One of the best overall measures for a binary classification model is 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, you must generate prediction probabilities for the test set. These probabilities are estimates for each of the classes, or answers, the model can predict, then we generate an ROC AUC score from the probabilities using scikit-learn's roc_auc_score method. The output from the score method reflects how many of the items in the test set the model predicted correctly. This score is skewed by the fact that the dataset the model was trained and tested with contains many more rows representing on-time arrivals than rows representing late arrivals. Because of this imbalance in the data, you're more likely to be correct if you predict that a flight will be on time than if you predict that a flight will be late.

You can learn more about the model's behavior by generating a confusion matrix, also known as an error matrix. The confusion matrix quantifies the number of times each answer was classified correctly or incorrectly. Specifically, it quantifies the number of false positives, false negatives, true positives, and true negatives.

To improve accuracy even more for a classification model, include precision and recall. Suppose the model was presented with three on-time arrivals and three delayed arrivals, and that it correctly

predicted two of the on-time arrivals, but incorrectly predicted that two of the delayed arrivals would be on time. In this case, the precision would be 50% (two of the four flights it classified as being on time actually were on time), while its recall would be 67% (it correctly identified two of the three on-time arrivals).

This much tuning of the accuracy should be enough for achieving our objective.

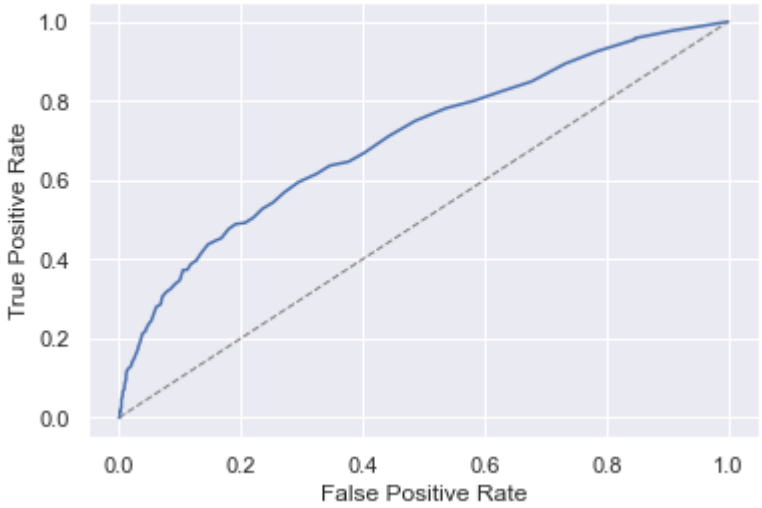
We execute a Matplotlib code in a new cell to plot the ROC curve for the machine-learning model we built, then we'll write a Python function that calls the machine-learning model we built to compute the likelihood that a flight will be on time.

Now we have an easy way to predict, with a single line of code, whether a flight is likely to be on time or late. Example: predict_delay('5/06/2018 20:25:00', 'MSP', 'ATL').

COMPARE WITH EXISTING MODEL

In this project we built the accuracy of our model in 4 incremental layers, while others have usually used one or two. The accuracy showed by our model is well beyond decent

```
Out[33]: Text(0, 0.5, 'True Positive Rate')
```



With this project will detect the pattern of delay from the airport level in which delays occur, give basic statistics on their magnitudes and frequencies. Besides, major airports in the United States will be under the microscope regarding the delay time, the major causes of delays, and the worst time of delays in comparison the month of the year/season.

More will be discussed in results and reports.

RESULTS

Calling 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:

```
In [24]: predicted = model.predict(test_x)
         model.score(test_x, test_y)

Out[24]: 0.8642634623943035
```

Generating an ROC AUC score from the probabilities using scikit-learn's roc_auc_score method:

```
In [26]: from sklearn.metrics import roc_auc_score
         probabilities = model.predict_proba(test_x)

In [27]: roc_auc_score(test_y, probabilities[:, 1])

Out[27]: 0.7014819895830565
```

Confusion matrix- The first column in that row shows how many flights were correctly predicted to be on time, while the second column reveals how many flights were predicted as delayed but weren't.

```
In [28]: from sklearn.metrics import confusion_matrix
         confusion_matrix(test_y, predicted)

Out[28]: array([[1903,   33],
                [ 272,   39]], dtype=int64)
```

Using the precision_score and recall_score methods for computing precision and recall:

```
In [29]: from sklearn.metrics import precision_score

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

Out[29]: 1.0

In [30]: from sklearn.metrics import recall_score

         recall_score(train_y, train_predictions)

Out[30]: 0.9992012779552716
```

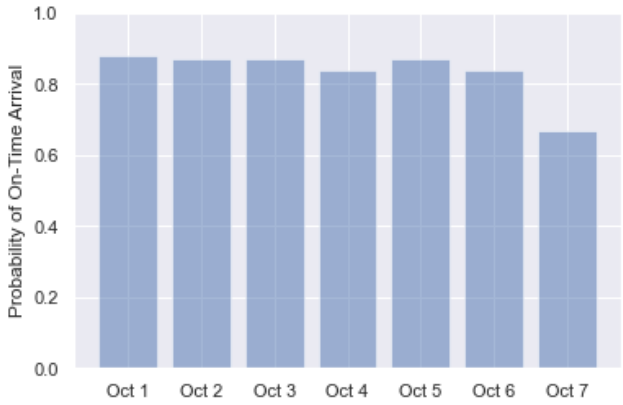
Final function for predicting the outcome:

```
In [35]: predict_delay('1/10/2018 21:45:00', 'JFK', 'ATL')

Out[35]: 0.88
```

Visualizing the output:

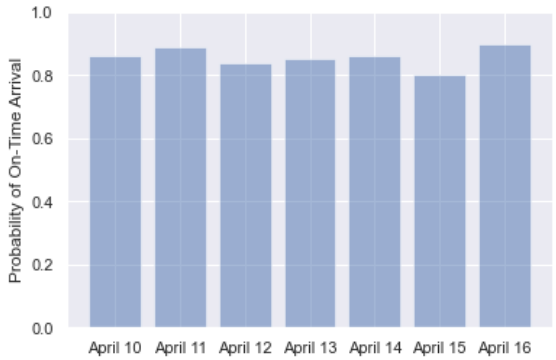
In [45]:
```
import numpy as np

labels = ('Oct 1', 'Oct 2', 'Oct 3', 'Oct 4', 'Oct 5', 'Oct 6', 'Oct 7')
values = (predict_delay('1/10/2018 21:45:00', 'JFK', 'ATL'),
          predict_delay('2/10/2018 21:45:00', 'JFK', 'ATL'),
          predict_delay('3/10/2018 21:45:00', 'JFK', 'ATL'),
          predict_delay('4/10/2018 21:45:00', 'JFK', 'ATL'),
          predict_delay('5/10/2018 21:45:00', 'JFK', 'ATL'),
          predict_delay('6/10/2018 21:45:00', 'JFK', 'ATL'),
          predict_delay('7/10/2018 21:45:00', 'JFK', 'ATL'))
alabels = np.arange(len(labels))

plt.bar(alabels, values, align='center', alpha=0.5)
plt.xticks(alabels, labels)
plt.ylabel('Probability of On-Time Arrival')
plt.ylim((0.0, 1.0))
```

Out[45]:
```
(0.0, 1.0)
```



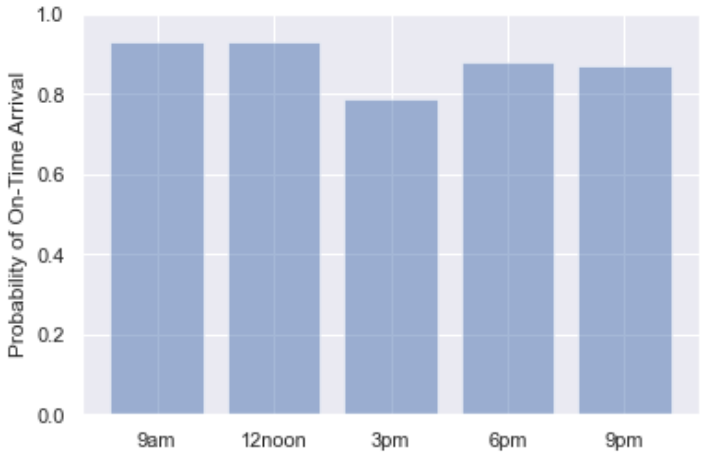
In [44]:
```
import numpy as np

labels = ('April 10', 'April 11', 'April 12', 'April 13', 'April 14', 'April 15', 'April 16')
values = (predict_delay('10/04/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('11/04/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('12/04/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('13/04/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('14/04/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('15/04/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('16/04/2018 13:00:00', 'JFK', 'MSP'))
alabels = np.arange(len(labels))

plt.bar(alabels, values, align='center', alpha=0.5)
plt.xticks(alabels, labels)
plt.ylabel('Probability of On-Time Arrival')
plt.ylim((0.0, 1.0))
```

Out[44]: (0.0, 1.0)



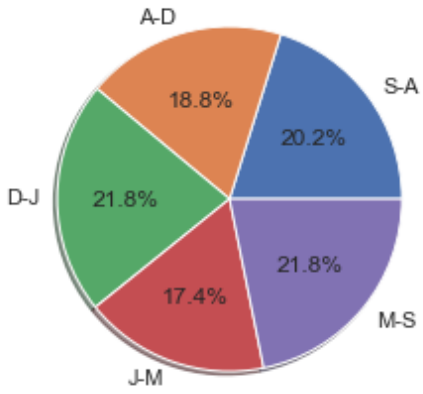
In [46]:
```
import numpy as np

labels = ('9am', '12noon', '3pm', '6pm', '9pm')
values = (predict_delay('30/01/2018 09:00:00', 'SEA', 'ATL'),
          predict_delay('30/01/2018 12:00:00', 'SEA', 'ATL'),
          predict_delay('30/01/2018 15:00:00', 'SEA', 'ATL'),
          predict_delay('30/01/2018 18:00:00', 'SEA', 'ATL'),
          predict_delay('30/01/2018 21:00:00', 'SEA', 'ATL'))
alabels = np.arange(len(labels))

plt.bar(alabels, values, align='center', alpha=0.5)
plt.xticks(alabels, labels)
plt.ylabel('Probability of On-Time Arrival')
plt.ylim((0.0, 1.0))
```

Out[46]: (0.0, 1.0)



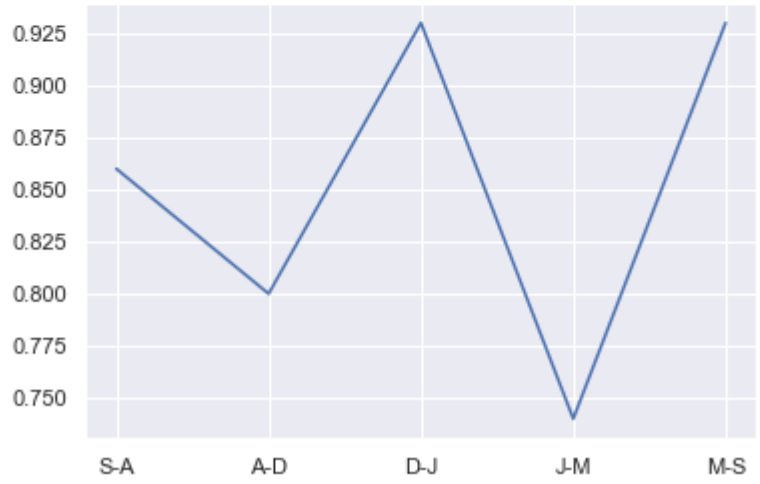
In [53]:
```
labels = ('S-A', 'A-D', 'D-J', 'J-M', 'M-S')
values = (predict_delay('01/01/2018 00:00:00', 'SEA', 'ATL'),
          predict_delay('01/01/2018 00:00:00', 'ATL', 'DTW'),
          predict_delay('01/01/2018 00:00:00', 'DTW', 'JFK'),
          predict_delay('01/01/2018 00:00:00', 'JFK', 'MSP'),
          predict_delay('01/01/2018 00:00:00', 'MSP', 'SEA'))
alabels = np.arange(len(labels))
plt.pie(values, autopct='%0.01f%%', shadow=True, labels=labels)
```

Out[53]:
```
([<matplotlib.patches.Wedge at 0xc67b0e8>,
  <matplotlib.patches.Wedge at 0xc67b580>,
  <matplotlib.patches.Wedge at 0xc67b9e8>,
  <matplotlib.patches.Wedge at 0xc67be68>,
  <matplotlib.patches.Wedge at 0xc68a2f8>],
 [Text(0.8860887187648293, 0.6518027174519169, 'S-A'),
  Text(-0.31202784558537505, 1.0548168673183747, 'A-D'),
  Text(-1.0999700869759472, 0.008112198106999612, 'D-J'),
  Text(-0.3736802785866346, -1.034583514944741, 'J-M'),
  Text(0.8512754342867517, -0.69665639664034, 'M-S')],
 [Text(0.4833211193262704, 0.35552875497377284, '20.2%'),
  Text(-0.17019700668293183, 0.5753546549009315, '18.8%'),
  Text(-0.5999836838050621, 0.004424835331090697, '21.8%'),
  Text(-0.20382560650180068, -0.5643182808789496, '17.4%'),
  Text(0.46433205506550085, -0.3799943981674581, '21.8%')])
```



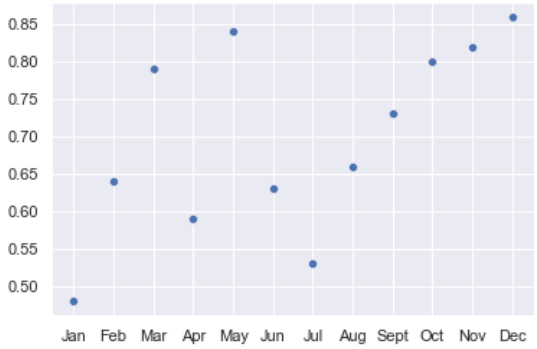
In [55]:
```
labels = ('S-A', 'A-D', 'D-J', 'J-M', 'M-S')
values = (predict_delay('01/01/2018 00:00:00', 'SEA', 'ATL'),
          predict_delay('01/01/2018 00:00:00', 'ATL', 'DTW'),
          predict_delay('01/01/2018 00:00:00', 'DTW', 'JFK'),
          predict_delay('01/01/2018 00:00:00', 'JFK', 'MSP'),
          predict_delay('01/01/2018 00:00:00', 'MSP', 'SEA'))
sns.lineplot(x=labels,y=values)
```

Out[55]: <AxesSubplot:>



In [57]:
```
labels = ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sept', 'Oct', 'Nov', 'Dec')
values = (predict_delay('01/01/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('01/02/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('01/03/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('01/04/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('01/05/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('01/06/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('01/07/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('01/08/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('01/09/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('01/10/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('01/11/2018 13:00:00', 'JFK', 'MSP'),
          predict_delay('01/12/2018 13:00:00', 'JFK', 'MSP'))
sns.scatterplot(x=labels, y=values)
```

Out[57]: <AxesSubplot:>



REPORT

ROC curve generated with Matplotlib: 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 your model. More importantly, the fact that this chart appears at all demonstrates that you can use Matplotlib in a Jupyter notebook.

The function takes input as 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 to compute the probability.

After experimenting with the function, we see that

In the first Bar graph – we can see the delay probabilities for the starting dates of October concerning the flights going from JFK to ATL

In the second Bar graph – we can see the delay probabilities for the mid-days of April concerning the flights going from JFK to MSP

In the third Bar graph – we can see the delay probabilities for the peak hours of 30th January concerning the flights going from SEA to ATL

The Pie chart shows the probabilities of flight delays during the New Years Eve concerning flights going in different routes throughout the Unites States.

The sns Line plot shows a more articulate visual representation of the New Years Eve data.

The scatterplot shows the probabilities of flight delays during the 1st of every month concerning the flights going from JFK to MSP.

We can view the probability for any date, any time concerning any flight taking the specified route.

CONCLUSION

Flight delays are an important subject in the literature due to their economic and environmental impacts. They may increase ticket costs to customers and operational costs to airlines. Apart from outcomes directly related to passengers, delay prediction is crucial during the decision-making process for every player in the air transportation system.

This paper will detect the pattern of delay from the airport level in which delays occur, give basic statistics on their magnitudes and frequencies. Besides, major airports in the United States will be under the microscope regarding the delay time, the major causes of delays, and the worst time of delays in comparison the month of the year/season

In this project we used Pandas to clean and prepare data, scikit-learn to build a machine-learning model, Matplotlib and seaborn to visualize the results.

REFERENCES

[1] Modern Data Visualization for Air Traffic Management By Frank Rehm, Frank Klawonn and Rudolf Kruse.

[2] Traffic Jam in the Air: Visualizing and Predicting US Airline Traffic Delays with Tableau and Exploratory By Mohamed Amir Omezzine, Wei Cao and ZhiYu Tian

[3] Why is your airplane often so late? - Unknown

[4] Visualization of Air Traffic during Covid19 Pandemic By C.Kuan

[5] Air Traffic Control using Big Data Analysis and Machine Learning By Prachi Dhariwal