Predicting Flight Delays
An Exploratory Data Analysis

Intro To Data Science - CS675

Introduction

- **Background:**
  - Importance of air travel and timely flights
  - Overview of the problem of flight delays

- **Study Objectives:**
  - Analyze flight delay data to identify key factors
  - Develop a predictive model for flight delays

- **Methodology Overview:**
  - Data cleaning, exploratory analysis, feature engineering, and model development

- **Study Significance:**
  - Potential impact on airlines and passengers
  - Benefits of understanding flight delay causes

Dataset

- **Data Source:**
  Publicly available data sourced from Kaggle. (https://www.kaggle.com/datasets/giovamata/airlinedelaycauses/code)

- **Structure of Dataset:**
  Contains flight information from 2008.
  Over 1.9 million records
  29 variables including flight number, carrier, departure and arrival delays, etc.

The Problem of Airline Delays

- Global Issue: Widespread issue, impacting millions of travelers daily.
- Economic Impact: Leads to lost productivity, increased airline costs, and traveler dissatisfaction.
- Common Causes: Weather, mechanical issues, air traffic control, and late-arriving aircraft.
- Late Aircraft Delay: A prominent delay cause - late arrival of the incoming aircraft for the next flight.
- Data Analysis: We aim to analyze the dataset to understand the underlying causes of these delays.

Data Loading and Cleaning

- **Data Loading:**

  Necessary libraries were imported: pandas, numpy, matplotlib, seaborn, and warnings.
  The dataset was loaded using pd.read_csv().

- **Data Cleaning:**

  - The 'Unnamed: 0' column was dropped.
  - Data shape and missing values were checked.
  - Rows with missing values in important columns were dropped.
  - Unneeded columns were removed: 'Year', 'Month', 'SecurityDelay', 'ArrTime', 'DayofMonth', 'CancellationCode'.

- **Correlation Analysis:**

  The correlation between columns was calculated using corr().

Data Loading and Cleaning

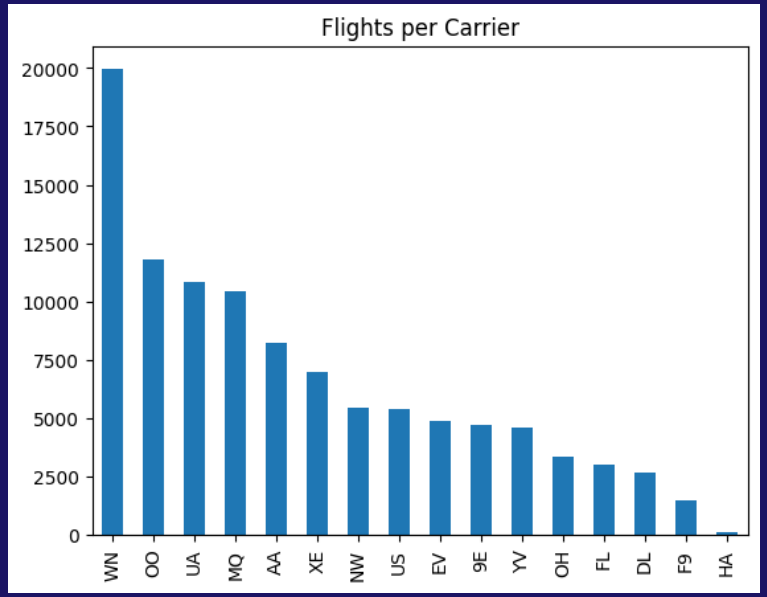
Descriptive Statistics

The dataset was reviewed to understand the general statistics of the data such as mean, median, min, max values.

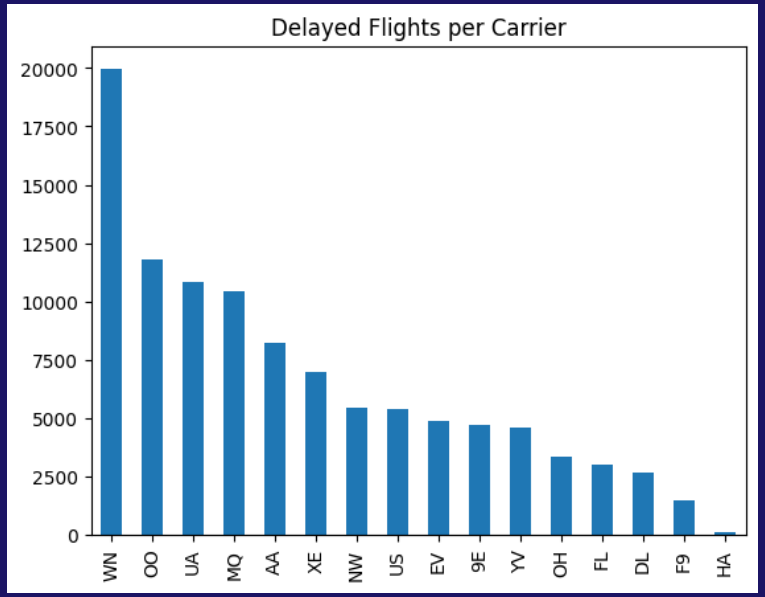
	DayOfWeek	DepTime	CRSDepTime	CRSArrTime	FlightNum	ActualElapsedTime	CRSElapsedTime	AirTime	ArrDelay	DepDelay	Distance	TaxiIn	TaxiOut	Cancelled	Diverted	CarrierDelay	WeatherDelay	NASDelay	LateAircraftDelay
count	103965.000000	103965.000000	103965.000000	103965.000000	103965.000000	103965.000000	103965.000000	103965.000000	103965.000000	103965.000000	103965.000000	103965.000000	103965.000000	103965.0	103965.0	103965.000000	103965.000000	103965.000000	103965.000000
mean	3.836281	1544.926706	1468.754052	1631.207887	2634.502333	126.479969	122.567143	99.486606	63.305218	59.392392	669.142837	7.176867	19.816496	0.0	0.0	19.664695	3.760708	14.007916	25.776810
std	1.909132	446.268135	419.794370	451.741484	2122.847356	66.487608	64.493118	63.490928	59.545949	58.121407	515.991601	6.040805	15.162228	0.0	0.0	42.905318	22.403702	34.027715	41.822383
min	1.000000	1.000000	25.000000	1.000000	1.000000	25.000000	19.000000	0.000000	15.000000	6.000000	24.000000	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000
25%	2.000000	1221.000000	1135.000000	1320.000000	744.000000	80.000000	77.000000	55.000000	26.000000	24.000000	316.000000	4.000000	11.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000
50%	4.000000	1604.000000	1510.000000	1700.000000	2016.000000	110.000000	105.000000	82.000000	43.000000	41.000000	529.000000	6.000000	16.000000	0.0	0.0	3.000000	0.000000	0.000000	9.000000
75%	5.000000	1915.000000	1815.000000	2009.000000	4268.000000	153.000000	149.000000	124.000000	79.000000	76.000000	861.000000	8.000000	24.000000	0.0	0.0	22.000000	0.000000	12.000000	34.000000
max	7.000000	2400.000000	2359.000000	2359.000000	7829.000000	693.000000	600.000000	589.000000	1357.000000	1355.000000	4502.000000	189.000000	383.000000	0.0	0.0	1120.000000	1049.000000	1357.000000	897.000000

Data Loading and Cleaning

Flights per Carrier: A bar plot was used to visualize the number of flights per unique carrier. This gave an insight into which airlines have the most flights.

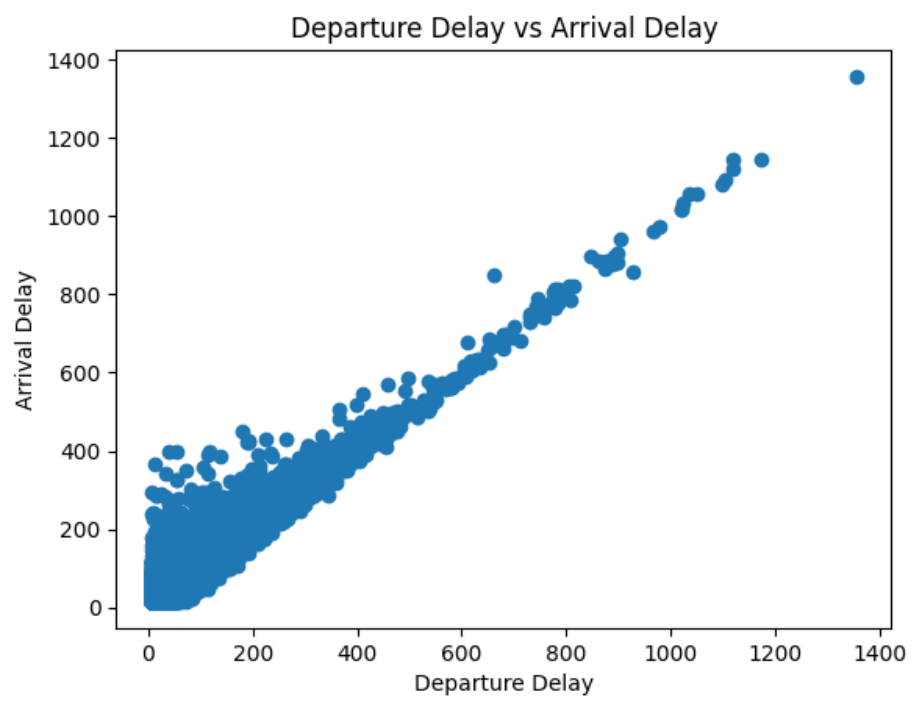


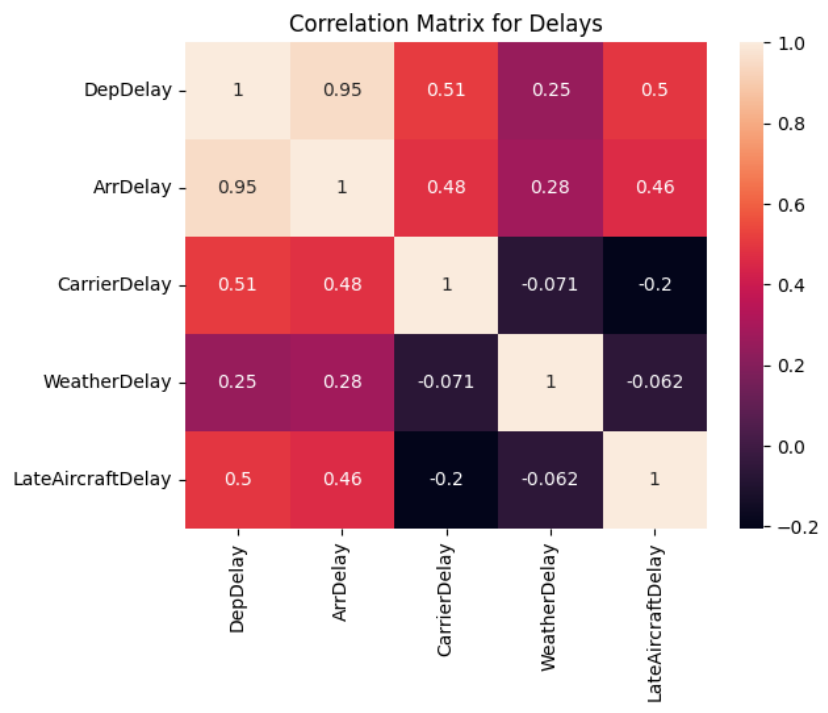
Average delay per Carrier: The average delay per carrier was calculated and visualized using a bar plot. This helped identify the carriers that have the highest average delays.



Data Loading and Cleaning

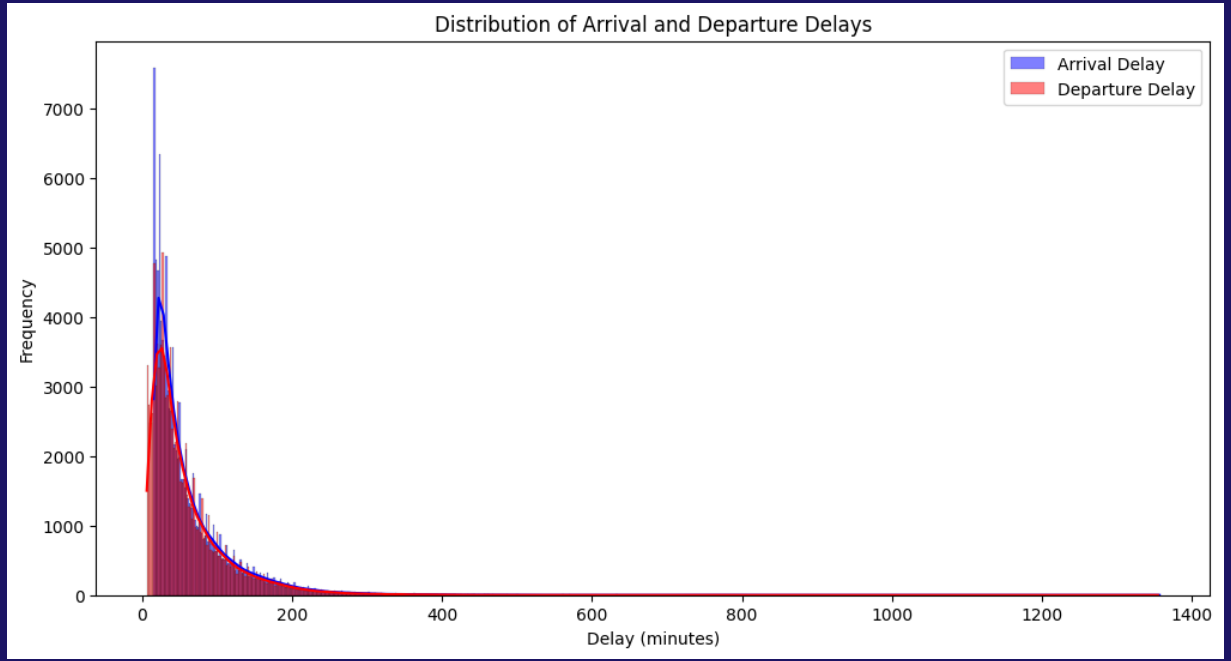
Scatter plots and Correlation matrix:





Data Loading and Cleaning

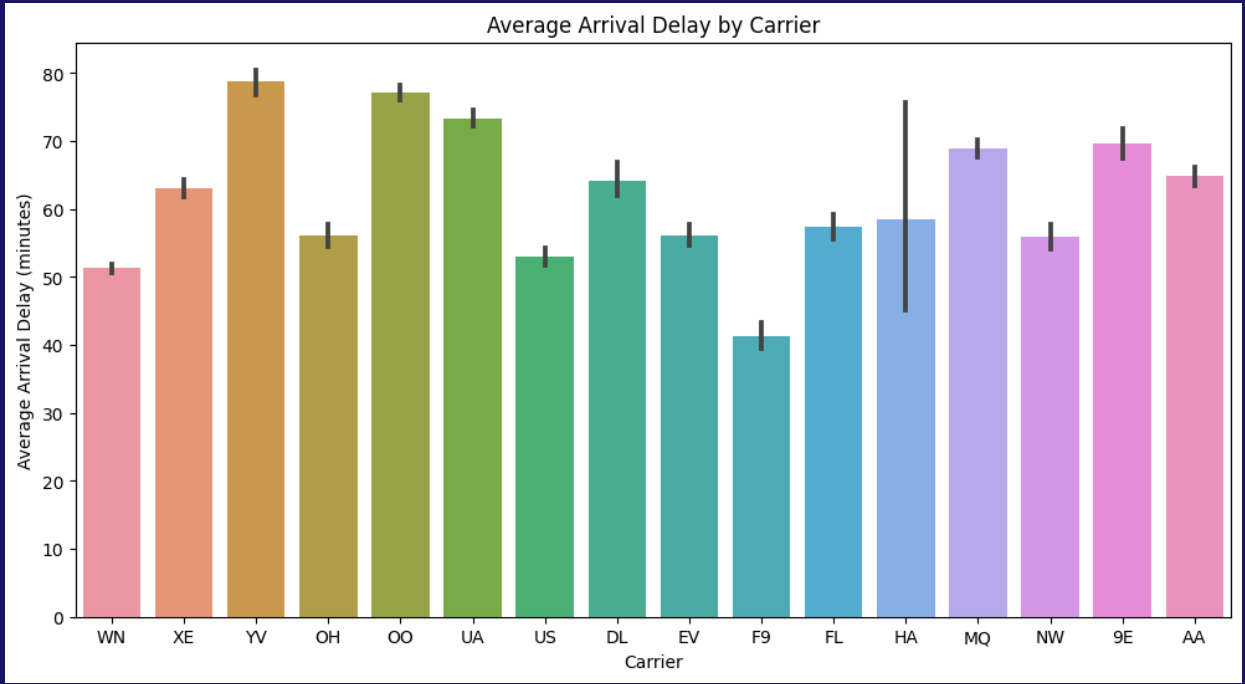
Distribution of Arrival and Departure Delays:
Histograms were used to visualize the distribution of arrival and departure delays. This helped in understanding the common delay times and identify any skewness in the data.



Data Loading and Cleaning

Average Arrival Delay by Carrier:

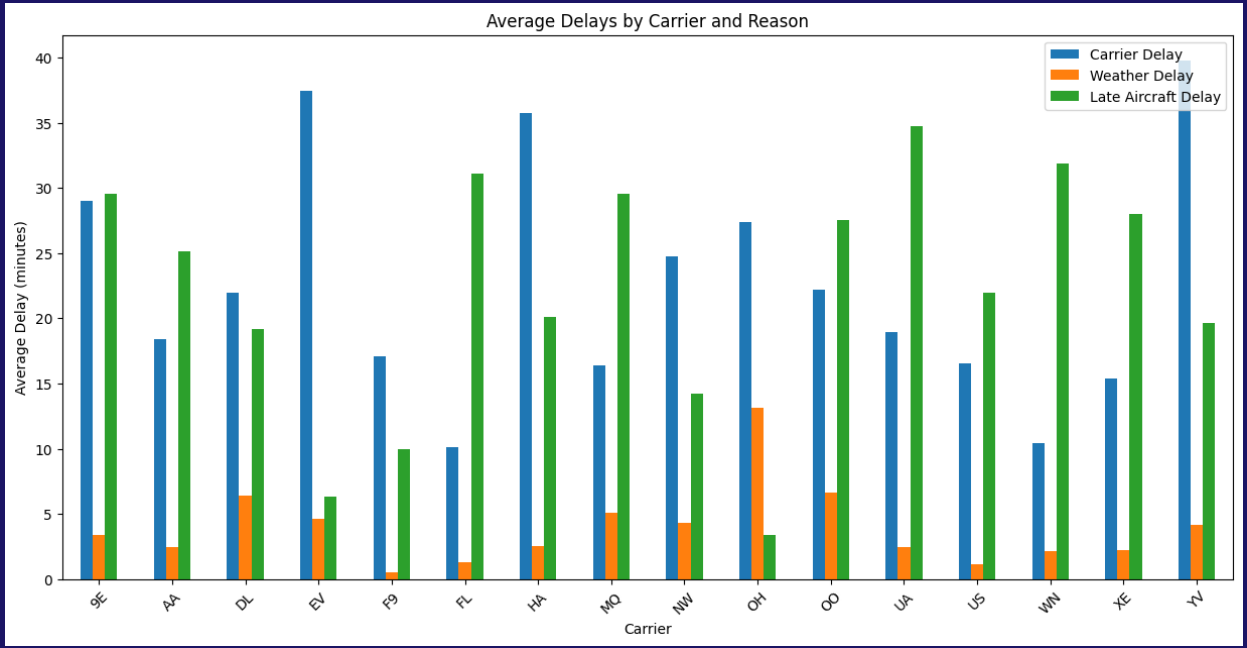
A bar plot was used to visualize the average arrival delay by carrier. This helped in identifying which carriers generally have more arrival delays.



Data Loading and Cleaning

Average Arrival Delay by Carrier:

A bar plot was used to visualize the average arrival delay by carrier. This helped in identifying which carriers generally have more arrival delays.



Feature Selection and Engineering

- **New Features**: Three new features were added to the dataset
  - **TotalDelay**: The sum of arrival and departure delays. This feature helps in capturing the total delay a flight has experienced.
  - **DepHour**: The hour of the scheduled departure time. This helps in understanding the distribution of delays across different hours of the day.
  - **TimeOfDay**: A categorical variable representing the time of day (Night, Morning, Afternoon, Evening) based on the scheduled departure time. This feature helps in understanding the distribution of delays across different periods of the day.

```
# Create a "TotalDelay" feature by combining departure and arrival delays
fd_ds['TotalDelay'] = fd_ds['ArrDelay'] + fd_ds['DepDelay']

# Create categorical features for specific hours of the day
fd_ds['DepHour'] = fd_ds['CRSDepTime'] // 100
fd_ds['TimeOfDay'] = pd.cut(fd_ds['DepHour'], bins=[-1, 5, 11, 17, 23], labels=['Night', 'Morning', 'Afternoon', 'Evening'])

# Display the updated dataset
fd_ds.head()
```

	DayOfWeek	DepTime	CRSDepTime	CRSArrTime	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	CRSElapsedTime	AirTime	...	TaxiOut	Cancelled	Diverted	CarrierDelay	WeatherDelay	NASDelay	LateAircraftDelay	TotalDelay	DepHour	TimeOfDay
3	4	1829.0	1755	1925.0	WN	3920.0	N464WN	90.0	90.0	77.0	...	10.0	0.0	0.0	2.0	0.0	0.0	32.0	68.0	17	Afternoon
5	4	1937.0	1830	1940.0	WN	509.0	N763SW	240.0	250.0	230.0	...	7.0	0.0	0.0	10.0	0.0	0.0	47.0	124.0	18	Evening
7	4	1644.0	1510	1725.0	WN	1333.0	N334SW	121.0	135.0	107.0	...	8.0	0.0	0.0	8.0	0.0	0.0	72.0	174.0	15	Afternoon
9	4	1452.0	1425	1625.0	WN	675.0	N286WN	228.0	240.0	213.0	...	8.0	0.0	0.0	3.0	0.0	0.0	12.0	42.0	14	Afternoon
11	4	1323.0	1255	1510.0	WN	4.0	N674AA	123.0	135.0	110.0	...	9.0	0.0	0.0	0.0	0.0	0.0	16.0	44.0	12	Afternoon

5 rows × 26 columns

Feature Selection and Engineering

Outlier Detection and Removal:

The interquartile range (IQR) was calculated for the ArrDelay column to detect outliers. The lower and upper bounds for outliers were defined and the rows with ArrDelay values outside this range were removed. This step was important to ensure the predictive model is not affected by extreme values.

```
# Calculate the IQR for 'ArrDelay'
Q1 = fd_ds['ArrDelay'].quantile(0.25)
Q3 = fd_ds['ArrDelay'].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers from the dataset
fd = fd_ds[(fd_ds['ArrDelay'] >= lower_bound) & (fd_ds['ArrDelay'] <= upper_bound)]

# Display the updated dataset
fd.head()
```

	DayOfWeek	DepTime	CRSDepTime	CRSArrTime	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	CRSElapsedTime	AirTime	...	TaxiOut	Cancelled	Diverted	CarrierDelay	WeatherDelay	NASDelay	LateAircraftDelay	TotalDelay	DepHour	TimeOfDay
3	4	1829.0	1755	1925.0	WN	3920.0	N464WN	90.0	90.0	77.0	...	10.0	0.0	0.0	2.0	0.0	0.0	32.0	68.0	17	Afternoon
5	4	1937.0	1830	1940.0	WN	509.0	N763SW	240.0	250.0	230.0	...	7.0	0.0	0.0	10.0	0.0	0.0	47.0	124.0	18	Evening
7	4	1644.0	1510	1725.0	WN	1333.0	N334SW	121.0	135.0	107.0	...	8.0	0.0	0.0	8.0	0.0	0.0	72.0	174.0	15	Afternoon
9	4	1452.0	1425	1625.0	WN	675.0	N286WN	228.0	240.0	213.0	...	8.0	0.0	0.0	3.0	0.0	0.0	12.0	42.0	14	Afternoon
11	4	1323.0	1255	1510.0	WN	4.0	N674AA	123.0	135.0	110.0	...	9.0	0.0	0.0	0.0	0.0	0.0	16.0	44.0	12	Afternoon

5 rows × 26 columns

Feature Selection and Engineering

Visualizing Effect of Outlier Removal:

Box plots were used to visualize the effect of outlier removal on the ArrDelay feature. The box plots showed a clear reduction in extreme values after outlier removal, making the data more suitable for modeling.

```
# Calculate the IQR for 'ArrDelay'
Q1 = fd_ds['ArrDelay'].quantile(0.25)
Q3 = fd_ds['ArrDelay'].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers from the dataset
fd = fd_ds[(fd_ds['ArrDelay'] >= lower_bound) & (fd_ds['ArrDelay'] <= upper_bound)]

# Display the updated dataset
fd.head()
```

	DayOfWeek	DepTime	CRSDepTime	CRSArrTime	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	CRSElapsedTime	AirTime	...	TaxiOut	Cancelled	Diverted	CarrierDelay	WeatherDelay	NASDelay	LateAircraftDelay	TotalDelay	DepHour	TimeOfDay
3	4	1829.0	1755	1925.0	WN	3920.0	N464WN	90.0	90.0	77.0	...	10.0	0.0	0.0	2.0	0.0	0.0	32.0	68.0	17	Afternoon
5	4	1937.0	1830	1940.0	WN	509.0	N763SW	240.0	250.0	230.0	...	7.0	0.0	0.0	10.0	0.0	0.0	47.0	124.0	18	Evening
7	4	1644.0	1510	1725.0	WN	1333.0	N334SW	121.0	135.0	107.0	...	8.0	0.0	0.0	8.0	0.0	0.0	72.0	174.0	15	Afternoon
9	4	1452.0	1425	1625.0	WN	675.0	N286WN	228.0	240.0	213.0	...	8.0	0.0	0.0	3.0	0.0	0.0	12.0	42.0	14	Afternoon
11	4	1323.0	1255	1510.0	WN	4.0	N674AA	123.0	135.0	110.0	...	9.0	0.0	0.0	0.0	0.0	0.0	16.0	44.0	12	Afternoon

5 rows × 26 columns

Overview of Machine Learning Algorithms

Logistic Regression

Supervised learning algorithm for binary classification tasks.

Models the probability of an event based on input features.

Decision Tree Model

Decision tree model had lower accuracy compared to other algorithms.

Model improvements are required to enhance its predictive capabilities.

Deep Neural Networks (DNN)

Supervised learning algorithm with interconnected layers of neurons.

Can model complex relationships between features and labels.

Logistic Regression

Preprocessed data using a column transformer to scale numerical features and one-hot encode categorical features.

```
# Create a binary variable for delays (1 if ArrDelay >= 30 minutes, 0 otherwise)
fd['Delayed'] = (fd['ArrDelay'] >= 30).astype(int)

# Create a binary variable for cancellations (1 if Cancelled == 1, 0 otherwise)
fd['Cancelled'] = fd['Cancelled'].astype(int)

# Define the features and target variable
features = ['DayOfWeek', 'UniqueCarrier', 'Origin', 'Dest', 'DepHour']
target = 'Delayed'

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(fd[features], fd[target], test_size=0.2, random_state=42)

# Preprocess the data
numerical_features = ['DepHour']
categorical_features = [ 'DayOfWeek', 'UniqueCarrier', 'Origin', 'Dest']

preprocessor = ColumnTransformer(transformers=[
    ('num', StandardScaler(), numerical_features),
    ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
])
```

Logistic Regression

- Created a logistic regression model using the preprocessed data.
- Trained the model on the training set and made predictions on the test set.
- Evaluated the model's performance using accuracy and classification report.

```
# Create a logistic regression model
logreg_model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(random_state=42))
])

# Train the logistic regression model
logreg_model.fit(X_train, y_train)

# Predict on the test set
y_pred_logreg = logreg_model.predict(X_test)

# Evaluate the logistic regression model
logreg_accuracy = accuracy_score(y_test, y_pred_logreg)
logreg_report = classification_report(y_test, y_pred_logreg)

print("Logistic Regression Accuracy:", logreg_accuracy)
print("Logistic Regression Classification Report:\n", logreg_report)
```

Logistic Regression

The significance of the Logistic Regression model results can be summarized as follows:

Accuracy: The model achieved an accuracy of 66.4%, indicating its overall predictive performance.

Logistic Regression Accuracy: 0.6643652102225887
Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	0.53	0.07	0.12	6566
1	0.67	0.97	0.79	12842
accuracy			0.66	19408
macro avg	0.60	0.52	0.46	19408
weighted avg	0.62	0.66	0.57	19408

Precision: The precision for delayed flights (class 1) was 67%, indicating the proportion of correctly predicted delayed flights.

Recall: The recall for delayed flights was 97%, indicating the proportion of actual delayed flights correctly identified by the model.

F1-score: The F1-score for delayed flights was 0.79, reflecting a balance between precision and recall.

Support: The dataset includes 6,566 non-delayed flights and 12,842 delayed flights.

Logistic Regression

Insights:

Overall, the Logistic Regression model shows moderate accuracy and performs better in predicting delayed flights.

However, there is room for improvement in correctly identifying non-delayed flights.

Logistic Regression Accuracy: 0.6643652102225887
Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	0.53	0.07	0.12	6566
1	0.67	0.97	0.79	12842
accuracy			0.66	19408
macro avg	0.60	0.52	0.46	19408
weighted avg	0.62	0.66	0.57	19408

Deep Neural Networks (DNN)

Preprocessed for Deep Neural Networks (DNN):

- The data was preprocessed for training a Deep Neural Network (DNN) using the preprocessor object.
- The preprocessor object, defined earlier, was used to preprocess the input features. It applied transformations such as scaling and one-hot encoding to the numerical and categorical features, respectively.

```
# Preprocess the data for the DNN
X_train_preprocessed = preprocessor.fit_transform(X_train)
X_test_preprocessed = preprocessor.transform(X_test)
```

- The training data X_train was preprocessed using the fit_transform method of the preprocessor object. This step learned the preprocessing parameters from the training data and transformed the data accordingly.

- The test data X_test was preprocessed using the transform method, which applied the learned preprocessing parameters to the test data without relearning them.

Deep Neural Networks (DNN)

Created the DNN Model:

- The DNN model was built using the Sequential class from the Keras API.
- It comprised four layers: three hidden layers with 64, 32, and 16 units respectively, and an output layer with a sigmoid activation function.
- The input_shape parameter of the first Dense layer was set to the number of preprocessed input features.

```
# Create the DNN model
dnn_model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train_preprocessed.shape[1],)),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

- ReLU activation was used in the hidden layers to capture complex patterns.
- The sigmoid activation function in the output layer facilitated binary classification.

Deep Neural Networks (DNN)

Compilation:
- The DNN model was compiled with 'adam' optimizer, 'binary_crossentropy' loss, and 'accuracy' metric.

Data Preparation:
- The sparse matrix was converted to a dense array using toarray().
- The data was split into training and validation sets.
- StandardScaler was used to standardize the input data.

Model Definition:
- Sequential model was created.
- Dense layers were added with appropriate units and activations.

Compilation and Training:
- Model was compiled again.
- Model was trained for 10 epochs with a batch size of 32 using the fit() function.

```
# Compile the DNN model
dnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Convert the Scipy sparse matrix to a dense NumPy array
X_train_dense = X_train_preprocessed.toarray()

# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train_dense, y_train, test_size=0.2, random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)

# Define the DNN model
dnn_model = Sequential()
dnn_model.add(Dense(units=64, activation='relu', input_dim=X_train.shape[1]))
dnn_model.add(Dense(units=1, activation='sigmoid'))

# Compile the model
dnn_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the DNN model
dnn_model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_val, y_val))
```

Deep Neural Networks (DNN)

Insights:

- The DNN model accurately predicted the delay status of flights with an overall accuracy of 0.65.
- The classification report provides precision, recall, and F1-score for each class (delayed and non-delayed flights).
- The model achieved higher precision (0.68) and recall (0.89) for delayed flights, indicating its effectiveness in identifying delays.
- For non-delayed flights, the model had lower precision (0.46) and recall (0.18), suggesting room for improvement.

Deep Neural Networks:

	precision	recall	f1-score	support
0	0.46	0.18	0.26	6566
1	0.68	0.89	0.77	12842
accuracy			0.65	19408
macro avg	0.57	0.54	0.52	19408
weighted avg	0.60	0.65	0.60	19408

Accuracy: 0.6499896949711459

- The F1-scores provide a balanced assessment of the model's performance, with a higher score (0.77) for delayed flights.
- These results are valuable for decision-making in the airline industry, helping allocate resources and improve operational efficiency.

Decision Tree Model

- Performed a train-test split on the dataset using the train_test_split function from sklearn.model_selection module.
- Split the data into X_train, X_test, y_train, y_test.

```
X_train, X_test, y_train, y_test = train_test_split(fd[features], fd[target], test_size=0.2, random_state=42, stratify=fd[target])
X_train = pd.DataFrame(X_train, columns=features)
X_test = pd.DataFrame(X_test, columns=features)
```

- Used 80% of the data for training and 20% for testing.
- Set the random_state parameter to 42 for reproducibility.
- Stratified the split based on the target variable to maintain the class distribution.
- Created pandas DataFrames X_train and X_test with column names as features.

Decision Tree Model

- Imported DecisionTreeClassifier from sklearn.tree.
- Created a decision tree model using Pipeline.
- Trained the model using fit() with X_train and y_train.
- Predicted on the test set using predict() with X_test.
- Evaluated the model's performance with accuracy_score() and classification_report().
- Printed the accuracy and classification report.

```
from sklearn.tree import DecisionTreeClassifier

# Create a decision tree model
dtree_model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', DecisionTreeClassifier(random_state=42))
])

# Train the decision tree model
dtree_model.fit(X_train, y_train)

# Predict on the test set
y_pred_dtree = dtree_model.predict(X_test)

# Evaluate the decision tree model
dtree_accuracy = accuracy_score(y_test, y_pred_dtree)
dtree_report = classification_report(y_test, y_pred_dtree)

print("Decision Tree Accuracy:", dtree_accuracy)
print("Decision Tree Classification Report:\n", dtree_report)
```

Decision Tree Model

Decision Tree Model Evaluation

- Decision Tree Accuracy: 0.58
- Precision for class 0 (no delay): 0.39
- Precision for class 1 (delay): 0.69
- Recall for class 0: 0.43
- Recall for class 1: 0.66
- F1-score for class 0: 0.41
- F1-score for class 1: 0.67
- Macro average F1-score: 0.54
- Weighted average F1-score: 0.58

```
Decision Tree Accuracy: 0.5794002473206925
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.39      0.43      0.41      6563
           1       0.69      0.66      0.67     12845

    accuracy                           0.58     19408
   macro avg       0.54      0.54      0.54     19408
weighted avg       0.59      0.58      0.58     19408
```

Decision Tree Model

Insights:

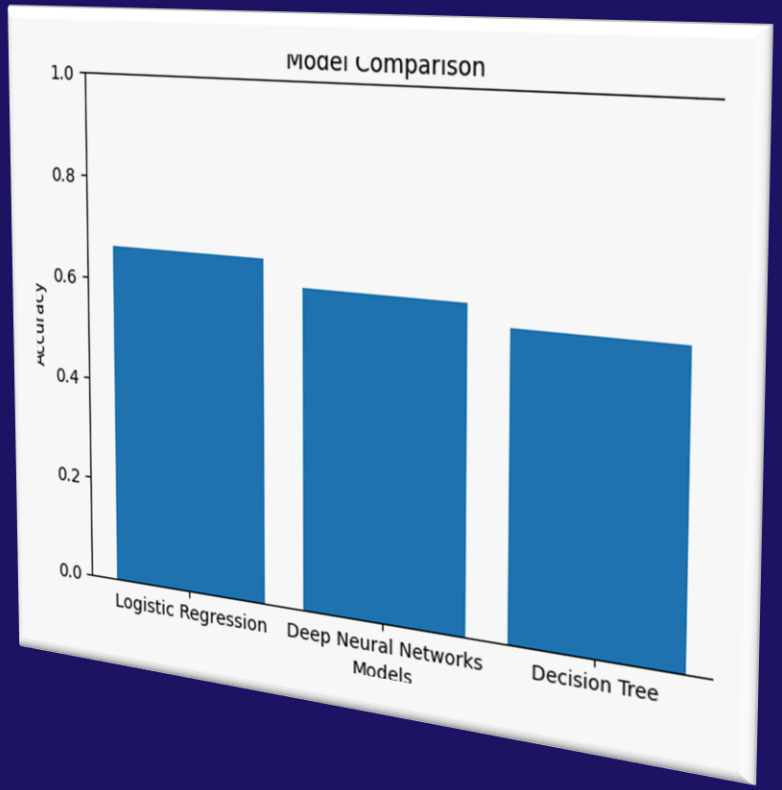
- The decision tree model shows moderate accuracy in predicting flight delays.
- It performs better in predicting delays (class 1) than no delays (class 0).
- Further improvements are needed to enhance its performance and accuracy.

Decision Tree Accuracy: 0.5794002473206925
Decision Tree Classification Report:

	precision	recall	f1-score	support
0	0.39	0.43	0.41	6563
1	0.69	0.66	0.67	12845
accuracy			0.58	19408
macro avg	0.54	0.54	0.54	19408
weighted avg	0.59	0.58	0.58	19408

Comparison

- **Logistic Regression** achieved the highest accuracy but has challenges predicting class 0 (no delay).
- **Deep Neural Networks** demonstrated higher accuracy than the decision tree but also faces challenges predicting class 0.
- **The decision tree model** has the lowest accuracy and requires further improvement to enhance its predictive capabilities.



Conclusion

Each model has its strengths and weaknesses, and the choice of the most suitable model depends on the specific requirements and characteristics of the dataset problem.

Further analysis and exploration may be needed to improve the performance of the models and address their limitations.

ThankYou

Presented By:
Pruthvi Raj Pudi
Vamshi Krishna Bangaru
Othniel Adu Mensah