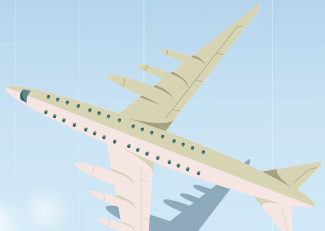
THE IMPACT OF AIRCRAFT WILDLIFE STRIKES ON PLANES

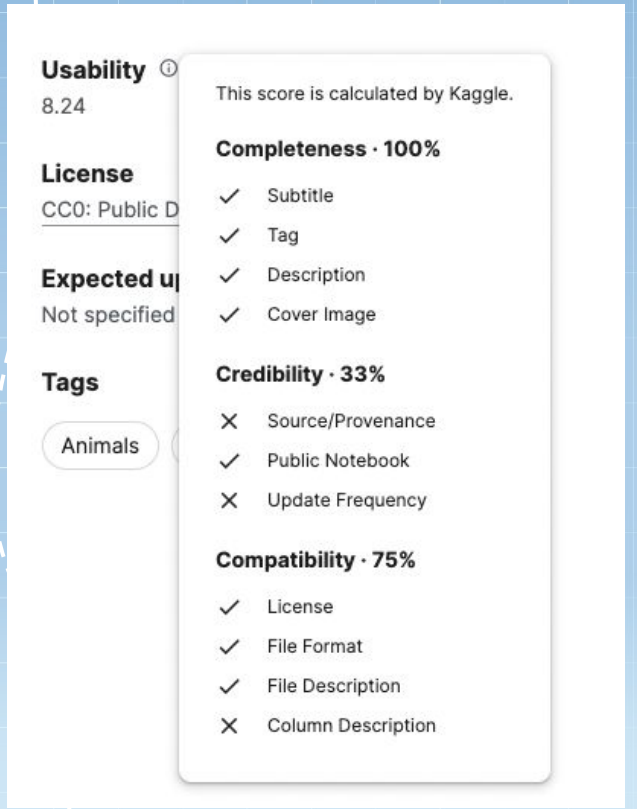
By Serena :)

THE PROBLEM

The problem of wildlife strikes in aviation poses a significant safety risk to both passengers and aircraft operations. Collisions between aircraft and birds or other wildlife can result in engine damage, malfunctions, emergency situations, and, in rare cases, catastrophic accidents. Identifying and mitigating these risks are paramount for aviation safety. Machine learning plays a crucial role in addressing this challenge by leveraging data analytics and predictive modeling.



STEP 1: IDENTIFYING A GOOD DATASET + DOWNLOADING



Dataset Overview:

- Name: Aircraft Wildlife Strike Dataset
- Usability: 8.24 (indicating its quality and usefulness)
- Timeframe: 1990 to 2015

Content:

- Contains records of wildlife strikes involving military, commercial, or civil aircraft.

Each record includes the following information:

- Incident date
- Aircraft operator
- Aircraft make and model
- Engine make and model
- Airport name and location
- Species name and quantity involved in the strike
- Aircraft damage information

Data Source:

- Compiled from reports received from airports, airlines, and pilots.
- The Federal Aviation Administration (FAA) as a reliable source of aviation safety data.

GOOGLE COLAB

Google Colab is a free, cloud-based platform that allows users to write and execute Python code in a collaborative and convenient online environment, making it particularly useful for data science and machine learning tasks.

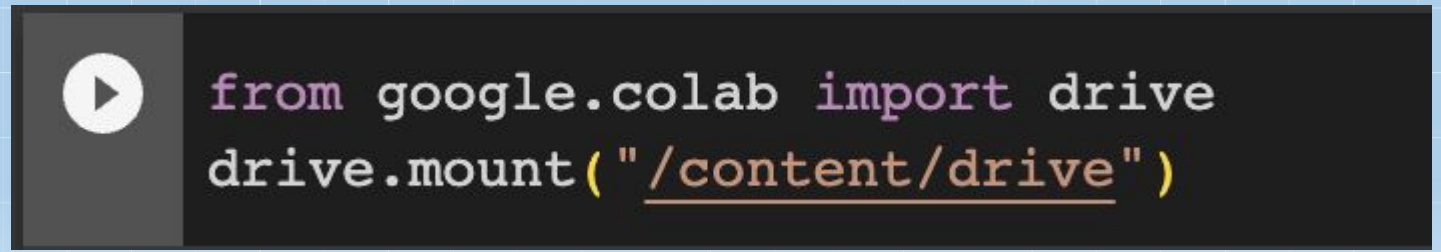
STEP 2: IMPORTING THE DATASET

1. Start by uploading your dataset to Google Drive.

2. In Google Colab, establish a connection to your Google Drive by navigating to the "Files" menu and clicking on the Google Drive icon.

3. Grant Colab access to your Google Drive.

4. Now, you can proceed to write the following piece of code.



THE LIBRARIES YOU NEED (ML)

01

SEABORN

Seaborn is a Python data visualization library that simplifies the creation of attractive and informative statistical graphics.

02

NUMPY

Numpy short for "Numerical Python," is a fundamental Python library for numerical and mathematical operations, providing support for multi-dimensional arrays

03

PANDAS

Pandas is a Python library used for data manipulation and analysis, offering data structures like DataFrames for storing and working with structured data, as well as various functions for tasks such as data cleaning, transformation, and exploration.

04

MATPLOTLIB

Matplotlib is a Python library for creating static, animated, and interactive visualizations in 2D and 3D. It offers a wide range of plotting capabilities, including line plots, scatter plots, bar plots, histograms, and more, making it a fundamental tool for data visualization and scientific plotting in Python.

05

SKLEARN

Scikit-learn, often referred to as sklearn, is a comprehensive Python library for machine learning that includes tools for various tasks like classification, regression, clustering, dimensionality reduction, and more, making it an essential resource for developing and evaluating machine learning models in Python.

06

IMBLEARN

Imbalanced-learn (imblearn) is a Python library tailored for handling imbalanced datasets in machine learning, offering techniques like resampling and ensemble methods.

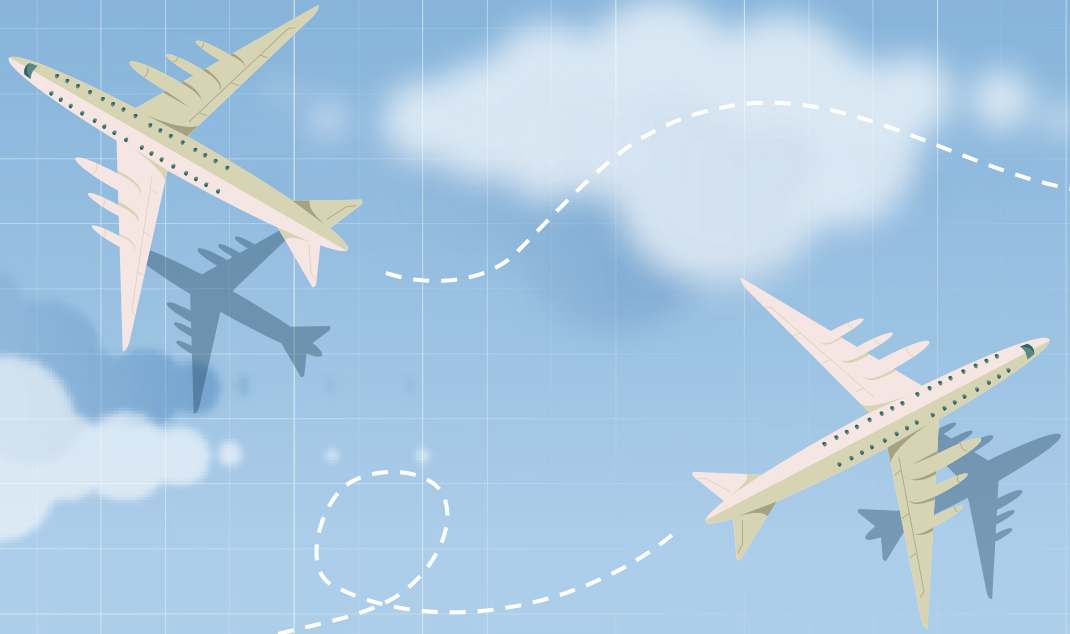
STEP 3: IMPORT ALL THE LIBRARIES

```
[ ]  import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt


     from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier, plot_tree
     from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import f1_score, ConfusionMatrixDisplay, roc_curve

     from tensorflow import keras
     from keras.models import Sequential
     from keras.layers import Input, Dense, Dropout
     from keras.optimizers import Adam
     from keras.callbacks import ReduceLROnPlateau, EarlyStopping

     from imblearn.over_sampling import RandomOverSampler, SMOTE
     from imblearn.under_sampling import RandomUnderSampler, TomekLinks
     from imblearn.combine import SMOTETomek
```

The script starts by importing various Python libraries, including Google Colab, NumPy, pandas, seaborn, matplotlib, scikit-learn (sklearn), TensorFlow and Keras for deep learning, and several modules from the imbalanced-learn library for dealing with imbalanced datasets.

WHOA!

THIS DATA NEEDS TO BE CLEANED!



01

DATA CLEANING

THERE'S A LOT TO DATA CLEAN

WHAT IS DATA CLEANING?

Data cleaning is the process of identifying and correcting errors and inconsistencies in datasets to improve data quality, involving tasks like handling missing values, removing duplicates, and addressing outliers.

```
target_cols = ['Engine1_Position', 'Engine2_Position', 'Engine3_Position', 'Engine4_Position']
data[target_cols] = data[target_cols].apply(pd.to_numeric) #CHange any stray strings to numeric values
```

```
data.drop(columns = ["Airport", "Operator", "Record ID", "Species Name"], inplace=True)
```

```
dummy_cols = ['Incident Year', 'Incident Month', 'Incident Day', 'Operator ID',
              'Aircraft', 'Aircraft Type', 'Aircraft Make', 'Aircraft Model', 'Engine Make', 'Engine Model',
              'Engine Type', 'Engine1_Position', 'Engine2_Position',
              'Engine3_Position', 'Engine4_Position', 'Airport ID', 'State',
              'FAA Region', 'Warning Issued', 'Flight Phase', 'Visibility',
              'Precipitation', 'Species ID', "Species Quantity"]
```

```
data.replace({"ENGINE SHUT DOWN" : "ENGINE SHUTDOWN"}, inplace=True)
```

```
data["Fatalities"].fillna(0, inplace=True)
data["Injuries"].fillna(0, inplace=True)
#data.dropna(inplace=True)
```

The code performs several data cleaning steps, including renaming columns, handling missing values, dropping unnecessary columns, and converting categorical variables into dummy variables.

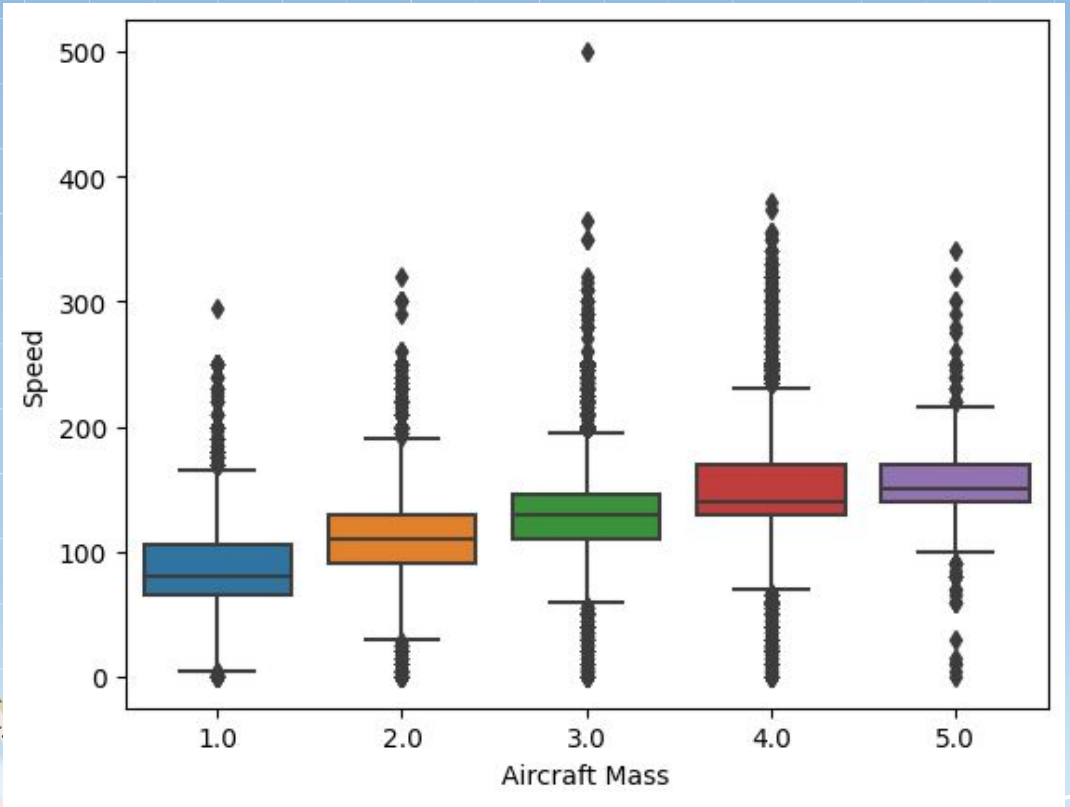
- Renaming certain columns in the DataFrame for consistency.
- Handling missing values in the "Fatalities" and "Injuries" columns by filling them with zeros.
- Dropping unnecessary columns ("Airport," "Operator," "Record ID," "Species Name").
- Converting categorical variables into dummy variables (one-hot encoding) for use in machine learning models.

02

PLOTTING

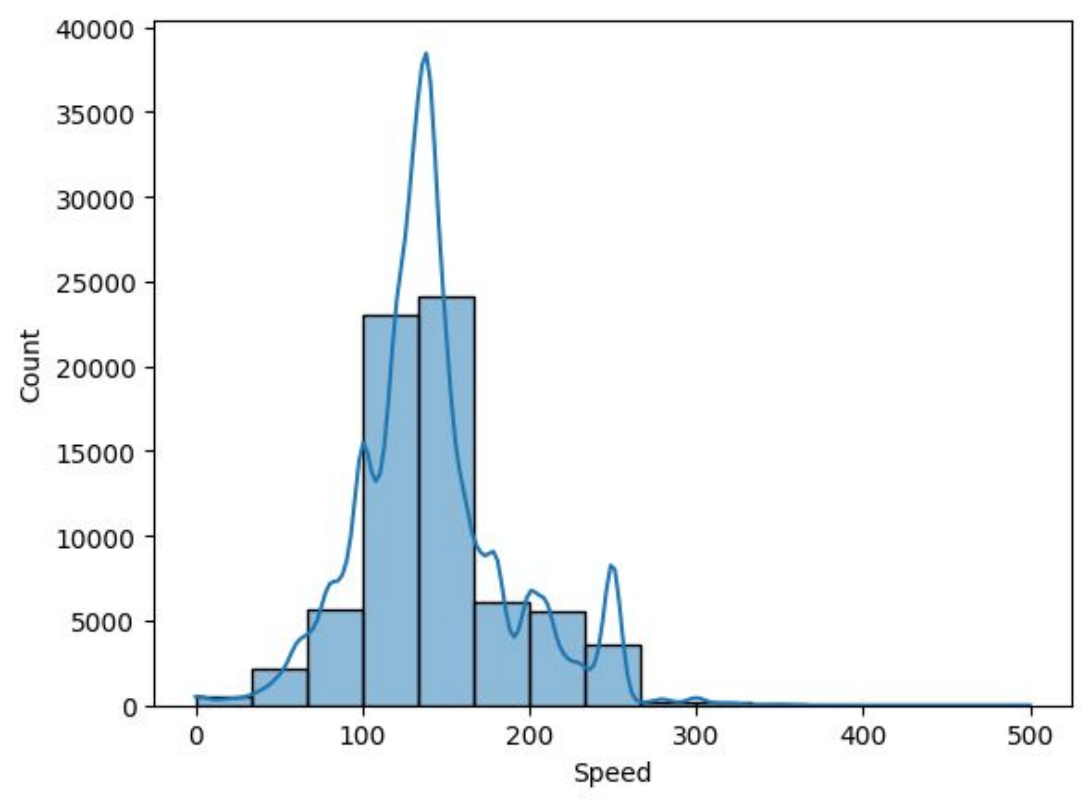
Using Seaborn

BOX PLOTS



A box and whisker plot or diagram (otherwise known as a boxplot), is a graph summarising a set of data. This boxplot shows how the data is distributed and it also shows any outliers.

HISTOGRAM PLOTS



A histogram is a graphical representation of a dataset's frequency distribution, with data points categorized into bins to show the count or relative frequency within each bin. It provides a visual summary of data distribution, aiding in the analysis of central tendencies and variations.

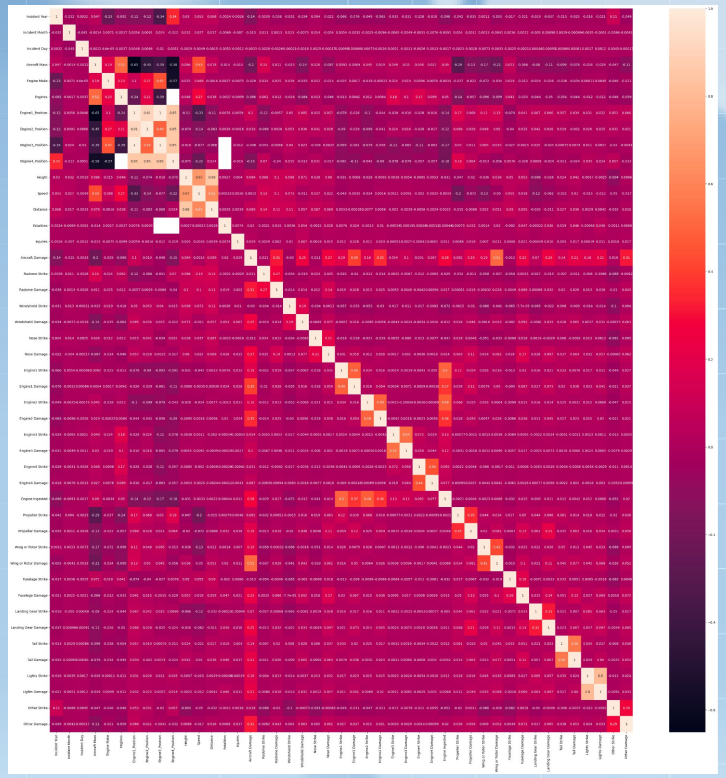
This histogram shows...

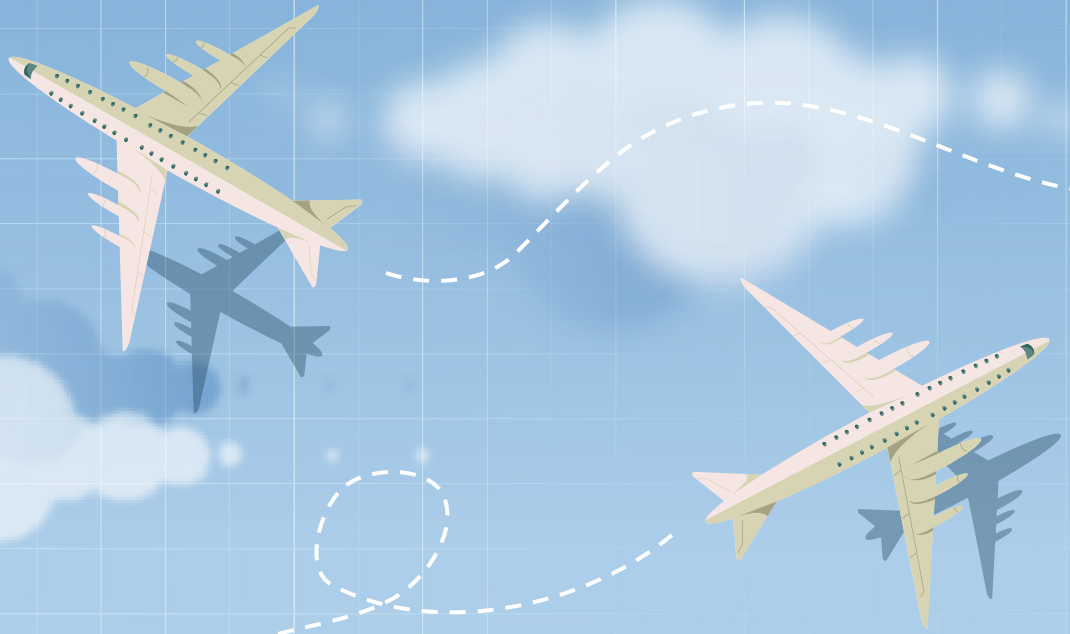
CORRELATION MATRIX

A correlation matrix shows the correlation between different variables (linear correlation). This matrix is useful for understanding connections between variables and is often visualized as a heatmap for easier interpretation.

\+ = positive slope (light colors)

\- = negative slope (dark colors)





03

MACHINE LEARNING

Using Sklearn

SPLITTING THE DATA

The dataset is split into training and testing sets using the train_test_split function from scikit-learn. 70% of the data is used for training, and 30% for testing.

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(data[X_columns], data[y_columns], train_size=0.7)
```

HANDLING THE IMBALANCED DATA

The SMOTETomek technique from the imblearn library is used to balance the class distribution in the training data. This technique combines Synthetic Minority Over-sampling Technique (SMOTE) and Tomek links to balance the dataset.

This takes a long time because our dataset is large

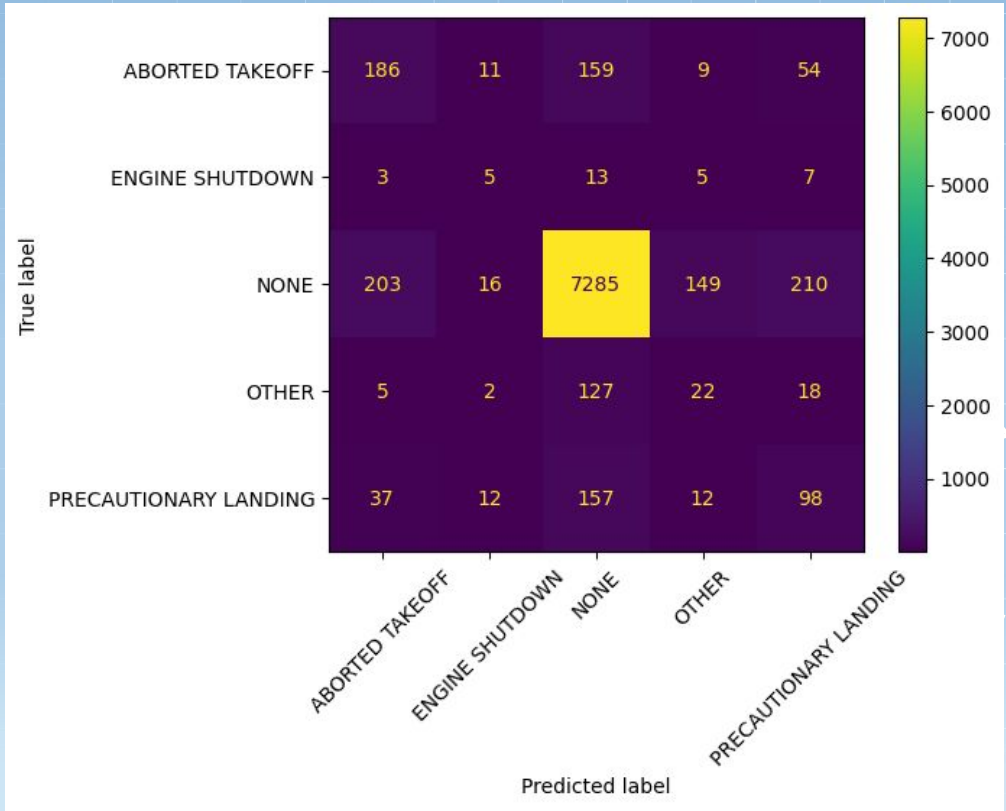
DECISION TREE CLASSIFIER

A Decision Tree Classifier model is created (tree_model) and trained on the balanced training data.
The accuracy of the model on the test data is printed.
The model's predictions on the test data are printed, and a confusion matrix is displayed.

```
[ ]  tree_model = DecisionTreeClassifier()
     tree_model.fit(X_train, y_train)
     print(tree_model.score(X_test, y_test))
     print(tree_model.predict(X_test))
     print(y_test)
```

MULTI-CLASS CONFUSION MATRIX - DECISION TREE

This is the fewest that were labeled None, meaning the minority category was more accurate, but the overall accuracy was lower.

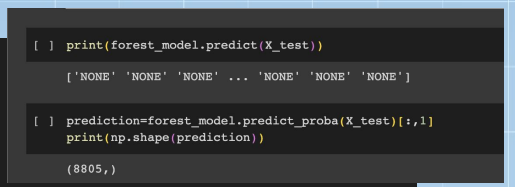


RANDOM FOREST CLASSIFIER

A Random Forest Classifier model is created (forest_model) with 100 estimators (trees) and trained on the balanced training data.
The accuracy of the model on the test data is printed.
The model's predictions on the test data are printed, and a confusion matrix is displayed.

```
[ ]  forest_model = RandomForestClassifier(n_estimators=100)
     forest_model.fit(X_train, y_train)
     print(forest_model.predict(X_test))
     print(forest_model.score(X_test, y_test))
```

```
     <ipython-input-25-034b45f8c267>:2: DataConversionWarning: A column-vector y wa
       forest_model.fit(X_train, y_train)
     ['NONE' 'NONE' 'NONE' ... 'NONE' 'NONE' 'NONE']
     0.9021010789324247
```

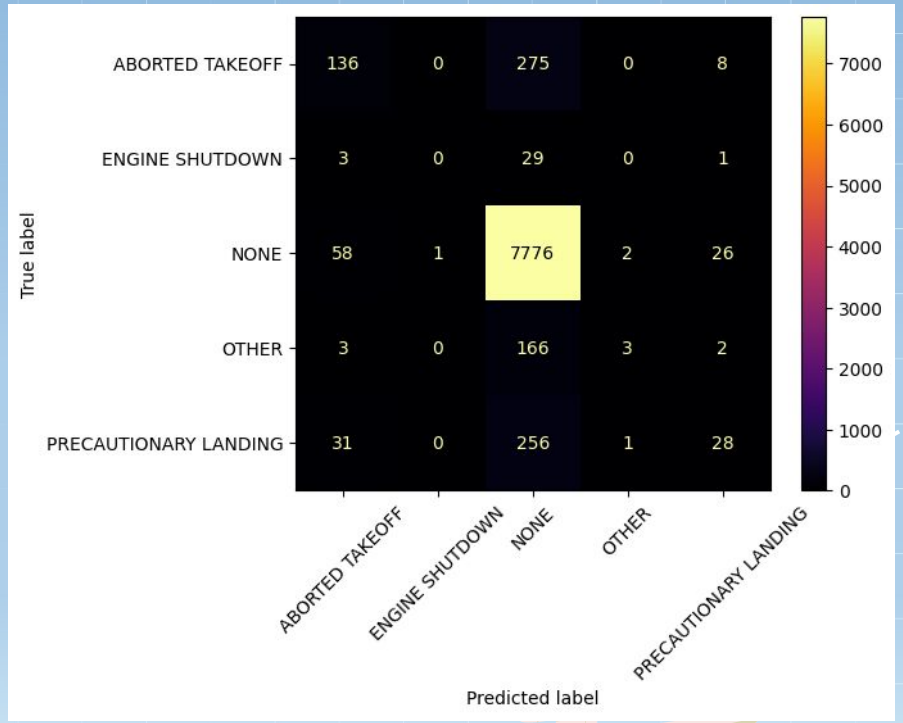


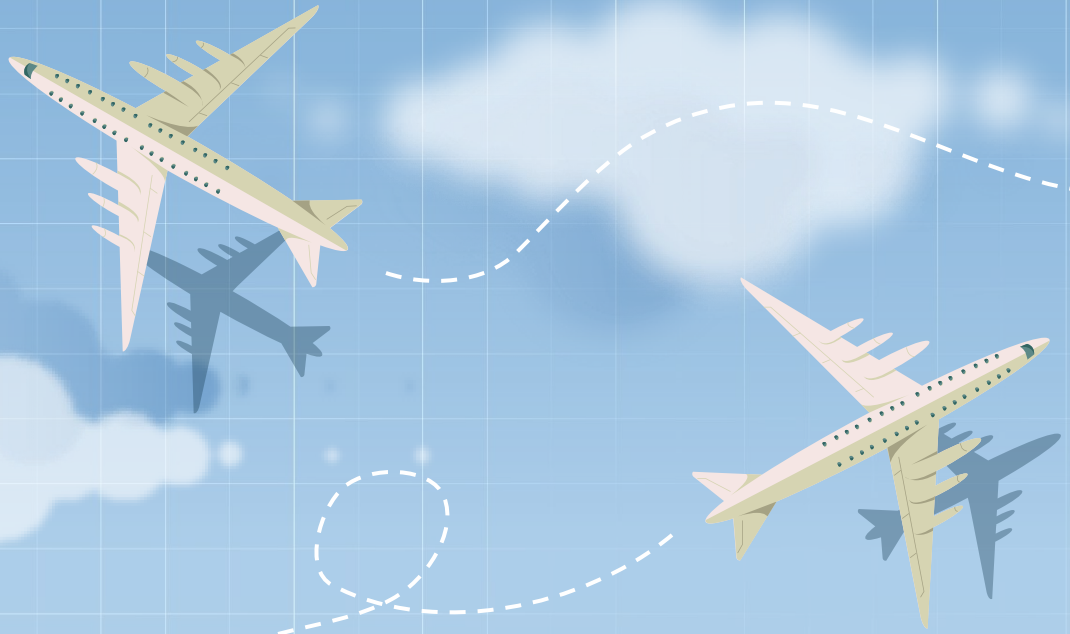
```
[ ] print(forest_model.predict(X_test))

    ['NONE' 'NONE' 'NONE' ... 'NONE' 'NONE' 'NONE']

[ ] prediction=forest_model.predict_proba(X_test)[:,1]
    print(np.shape(prediction))

    (8805,)
```

MULTI-CLASS CONFUSION MATRIX - RANDOM FOREST

This is the 2nd fewest that were labeled None, meaning the minority category was more accurate, but the overall accuracy was lower, however, not as low the decision tree.

THE MOST BALANCED (of the 3 results)





04

DEEP LEARNING

Using TensorFlow & Keras

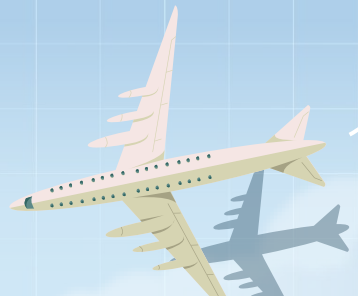
DNN MODEL

- A sequential deep neural network (DNN) model (dnn_model) is defined using Keras.
- The model architecture includes multiple hidden layers and dropout layers to prevent overfitting.
- The DNN model is trained using the balanced training data.
- The DNN model's predictions on the test data are compared to the true labels.
- Adam optimizer and categorical cross-entropy loss

```
dnn_model.add(Dense(100, activation = "relu"))
dnn_model.add(Dropout(0.5))

dnn_model.add(Dense(20, activation = "relu"))
dnn_model.add(Dropout(0.5))


dnn_model.add(Dense(len(y_test_dummies.columns), activation = "softmax")) #softmax is sigmoid for multiple classes. sigmoid for classifie
```

+ Code + Text

```
[ ] dnn_optimizer = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999)
    dnn_loss = "CategoricalCrossentropy"

    early_stopping = EarlyStopping(monitor= "val_loss", patience=10, restore_best_weights=True)
    plateau_monitor = ReduceLROnPlateau(monitor="val_loss", factor=0.1, patience=15, restore_best_weights=True)

[ ] dnn_model.compile(dnn_optimizer, dnn_loss)
    dnn_model.fit(X_train, y_train_dummies, batch_size=3000, epochs=100, verbose=1, validation_split=0.2, callbacks=[early_stopping, plateau_
```



MULTI-CLASS CONFUSION MATRIX - DNNS

This data cannot be trusted because we need more data from the minority categories to produce more reliable predictions.

AND all of the predictions are coming out with the same result → the machine learning is not learning anything.

Accuracy → ~ 0%