Multiple Classification Models for Flight Cancellations

In this project for Flyzy's predictive model of flight cancellations, we need to build, evaluate, and compare multiple classification models. The dataset provided has a variety of features that can influence flight cancellations, and our goal is to identify which model provides the best predictions based on several metrics.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, roc_auc_score
```

1. Load and Inspect Data

+ Code + Text

```
# Load the dataset
data = pd.read_csv('/content/Flyzy Flight Cancellation - Sheet1.csv')
```

```
# Inspect the data
print(data.info())
print(data.describe())
print(data.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Flight ID                     3000 non-null   int64
 1   Airline                       3000 non-null   object
 2   Flight_Distance               3000 non-null   int64
 3   Origin_Airport                3000 non-null   object
 4   Destination_Airport           3000 non-null   object
 5   Scheduled_Departure_Time      3000 non-null   int64
 6   Day_of_Week                   3000 non-null   int64
 7   Month                         3000 non-null   int64
 8   Airplane_Type                 3000 non-null   object
 9   Weather_Score                 3000 non-null   float64
 10  Previous_Flight_Delay_Minutes  3000 non-null   float64
 11  Airline_Rating                3000 non-null   float64
 12  Passenger_Load                3000 non-null   float64
 13  Flight_Cancelled              3000 non-null   int64
dtypes: float64(4), int64(6), object(4)
memory usage: 328.2+ KB
None
          Flight ID  Flight_Distance  Scheduled_Departure_Time  Day_of_Week  \
count  3.000000e+03      3000.000000               3000.000000  3000.000000
mean   4.997429e+06       498.909333                 11.435000     3.963000
std    2.868139e+06        98.892266                  6.899298     2.016346
min    3.681000e+03       138.000000                  0.000000     1.000000
25%    2.520313e+06       431.000000                  6.000000     2.000000
50%    5.073096e+06       497.000000                 12.000000     4.000000
75%    7.462026e+06       566.000000                 17.000000     6.000000
max    9.999011e+06       864.000000                 23.000000     7.000000

             Month  Weather_Score  Previous_Flight_Delay_Minutes  \
count  3000.000000    3000.000000                    3000.000000
mean      6.381000       0.524023                      26.793383
std       3.473979       0.290694                      27.874733
min       1.000000       0.000965                       0.000000
25%       3.000000       0.278011                       7.000000
50%       6.000000       0.522180                      18.000000
75%       9.000000       0.776323                      38.000000
max      12.000000       1.099246                     259.000000

       Airline_Rating  Passenger_Load  Flight_Cancelled
count     3000.000000     3000.000000       3000.000000
mean         2.317439        0.515885          0.690667
std          1.430386        0.295634          0.462296
min          0.000103        0.001039          0.000000
```

```
25%          1.092902       0.265793         0.000000
50%          2.126614       0.517175         1.000000
75%          3.525746       0.770370         1.000000
max          5.189038       1.123559         1.000000
   Flight ID    Airline  Flight_Distance Origin_Airport Destination_Airport  \
0    7319483  Airline D              475      Airport 3           Airport 2
1    4791965  Airline E              538      Airport 5           Airport 4
2    2991718  Airline C              565      Airport 1           Airport 2
3    4220106  Airline E              658      Airport 5           Airport 3
4    2263008  Airline E              566      Airport 2           Airport 2
```

2. Data Preprocessing

```
# Handle missing values
imputer = SimpleImputer(strategy='most_frequent')
data_imputed = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)
```

```
# Encode categorical variables
categorical_features = ['Airline', 'Origin_Airport', 'Destination_Airport', 'Airplane_Type']
data_encoded = pd.get_dummies(data_imputed, columns=categorical_features, drop_first=True)
```

```
# Ensure the target variable is numeric
data_encoded['Flight_Cancelled'] = pd.to_numeric(data_encoded['Flight_Cancelled'], errors='coerce')
```

```
# Drop rows with NaN values in the target variable after conversion
data_encoded = data_encoded.dropna(subset=['Flight_Cancelled'])
```

```
# Inspect columns to verify presence of 'Flight_ID' and 'Flight_Cancelled'
print("Columns in the DataFrame:", data_encoded.columns)
```

```
Columns in the DataFrame: Index(['Flight ID', 'Flight_Distance', 'Scheduled_Departure_Time',
       'Day_of_Week', 'Month', 'Weather_Score',
       'Previous_Flight_Delay_Minutes', 'Airline_Rating', 'Passenger_Load',
       'Flight_Cancelled', 'Airline_Airline B', 'Airline_Airline C',
       'Airline_Airline D', 'Airline_Airline E', 'Origin_Airport_Airport 2',
       'Origin_Airport_Airport 3', 'Origin_Airport_Airport 4',
       'Origin_Airport_Airport 5', 'Destination_Airport_Airport 3',
       'Destination_Airport_Airport 4', 'Destination_Airport_Airport 5',
       'Airplane_Type_Type B', 'Airplane_Type_Type C', 'Airplane_Type_Type D',
       'Airplane_Type_Type E'],
      dtype='object')
```

```
# Ensure 'Flight_ID' is present before dropping
if 'Flight_ID' in data_encoded.columns:
    X = data_encoded.drop(['Flight_ID', 'Flight_Cancelled'], axis=1)
else:
    X = data_encoded.drop(['Flight_Cancelled'], axis=1)

y = data_encoded['Flight_Cancelled']
```

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Standardize numerical features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

3. Build Advanced Models

```
# Build Decision Tree Model
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train_scaled, y_train)
```

```
▾        DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

```
# Predictions and Evaluation
y_pred_dt = dt_model.predict(X_test_scaled)
print("Decision Tree Classification Report:\n", classification_report(y_test, y_pred_dt))
print("Decision Tree ROC-AUC Score:", roc_auc_score(y_test, y_pred_dt))
```

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

           0       0.95      0.89      0.92       187
           1       0.95      0.98      0.97       413

    accuracy                           0.95       600
   macro avg       0.95      0.94      0.94       600
weighted avg       0.95      0.95      0.95       600

Decision Tree ROC-AUC Score: 0.9368388341469099
```

```
from sklearn.ensemble import RandomForestClassifier

# Build Random Forest Model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train_scaled, y_train)

# Predictions and Evaluation
y_pred_rf = rf_model.predict(X_test_scaled)
print("Random Forest Classification Report:\n", classification_report(y_test, y_pred_rf))
print("Random Forest ROC-AUC Score:", roc_auc_score(y_test, y_pred_rf))
```

```
Random Forest Classification Report:
               precision    recall  f1-score   support

           0       0.96      1.00      0.98       187
           1       1.00      0.98      0.99       413

    accuracy                           0.99       600
   macro avg       0.98      0.99      0.98       600
weighted avg       0.99      0.99      0.99       600

Random Forest ROC-AUC Score: 0.9903147699757869
```

```
from sklearn.linear_model import LogisticRegression

# Build Logistic Regression Model
lr_model = LogisticRegression(random_state=42)
lr_model.fit(X_train_scaled, y_train)

# Predictions and Evaluation
y_pred_lr = lr_model.predict(X_test_scaled)
print("Logistic Regression Classification Report:\n", classification_report(y_test, y_pred_lr))
print("Logistic Regression ROC-AUC Score:", roc_auc_score(y_test, y_pred_lr))
```

```
Logistic Regression Classification Report:
               precision    recall  f1-score   support

           0       0.71      0.61      0.66       187
           1       0.84      0.89      0.86       413

    accuracy                           0.80       600
   macro avg       0.77      0.75      0.76       600
weighted avg       0.80      0.80      0.80       600

Logistic Regression ROC-AUC Score: 0.7505859046237909
```

Insights

After building and evaluating the models, we will compare their performance metrics to decide on the best model. For the sake of brevity, the code for SVM, GBM, k-NN, and Neural Networks is not included but can be similarly implemented and evaluated.

By comparing the results, we can discuss the trade-offs and select the most suitable model for predicting flight cancellations. The selected model will then be used to generate actionable insights for Flyzy's business objectives.