# Airline Passenger Satisfaction Predictor

Nathan Yong, Lucas Zhang

# Objective

- Determine the whether customers will be satisfied with their airline based on factors given.
- Determine correlation between satisfaction factors and/or onboard/offboard services.
- Business applicable.

# Dataset Briefing

Customer satisfaction correlation to many other areas of off-flight and inflight experience

Categorical Data: Gender, Customer, Type of Travel, Class, Satisfaction

Numerical Data: Age, Flight Distance, Departure Delay in Minutes, Arrival Delay in Minutes

Numerical Data (Scale of 1-5): Inflight Wifi Service, Departure/Arrival Time Convenient, Ease of Online Booking, Gate Location, Food and Drink, Online Boarding, Seat Comfort, Inflight Entertainment, On-board service, Leg Room Service, Baggage Handling, Check-in Service, Inflight Service, Cleanliness

| | id | Gender | Customer Type | Age | Type of Travel | Class | Flight Distance | Inflight wifi service | Departure/Arrival time convenient | Ease of Online booking | ... | Inflight entertainment | On-board service | Leg room service | Baggage handling | Checkin service | Inflight service | Cleanliness | Departure Delay in Minutes | Arrival Delay in Minutes | satisfaction |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 70172 | Male | Loyal Customer | 13 | Personal Travel | Eco Plus | 460 | 3 | 4 | 3 | ... | 5 | 4 | 3 | 4 | 4 | 5 | 5 | 25 | 18.0 | neutral or dissatisfied |
| **1** | 5047 | Male | disloyal Customer | 25 | Business travel | Business | 235 | 3 | 2 | 3 | ... | 1 | 1 | 5 | 3 | 1 | 4 | 1 | 1 | 6.0 | neutral or dissatisfied |
| **2** | 110028 | Female | Loyal Customer | 26 | Business travel | Business | 1142 | 2 | 2 | 2 | ... | 5 | 4 | 3 | 4 | 4 | 4 | 5 | 0 | 0.0 | satisfied |
| **3** | 24026 | Female | Loyal Customer | 25 | Business travel | Business | 562 | 2 | 5 | 5 | ... | 2 | 2 | 5 | 3 | 1 | 4 | 2 | 11 | 9.0 | neutral or dissatisfied |
| **4** | 119299 | Male | Loyal Customer | 61 | Business travel | Business | 214 | 3 | 3 | 3 | ... | 3 | 3 | 4 | 4 | 3 | 3 | 3 | 0 | 0.0 | satisfied |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **99** | 94171 | Female | disloyal Customer | 23 | Business travel | Eco | 192 | 2 | 1 | 2 | ... | 2 | 3 | 1 | 4 | 2 | 3 | 2 | 3 | 0.0 | neutral or dissatisfied |
| **00** | 73097 | Male | Loyal Customer | 49 | Business travel | Business | 2347 | 4 | 4 | 4 | ... | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 0 | 0.0 | satisfied |
| **01** | 68825 | Male | disloyal Customer | 30 | Business travel | Business | 1995 | 1 | 1 | 1 | ... | 4 | 3 | 2 | 4 | 5 | 5 | 4 | 7 | 14.0 | neutral or dissatisfied |
| **02** | 54173 | Female | disloyal Customer | 22 | Business travel | Eco | 1000 | 1 | 1 | 1 | ... | 1 | 4 | 5 | 1 | 5 | 4 | 1 | 0 | 0.0 | neutral or dissatisfied |
| **03** | 62567 | Male | Loyal Customer | 27 | Business travel | Business | 1723 | 1 | 3 | 3 | ... | 1 | 1 | 1 | 4 | 4 | 3 | 1 | 0 | 0.0 | neutral or dissatisfied |

# Note on Datasets

```
df_train = pd.read_csv('./train.csv', index_col=[0])
df_test = pd.read_csv('./test.csv', index_col=[0])
```

# Data Preprocessing

- Null Values
- Standardization
- Encoding

```python
# Display the Null Values

display(df_train.isnull().sum())
display(df_test.isnull().sum())
display(df_train[pd.isnull(df_train).any(axis=1)])
```

```
Gender                                 0
Customer_Type                          0
Age                                    0
Type_of_Travel                         0
Class                                  0
Flight_Distance                        0
Inflight_wifi_service                  0
Departure/Arrival_time_convenient      0
Ease_of_Online_booking                 0
Gate_location                          0
Food_and_drink                         0
Online_boarding                        0
Seat_comfort                           0
Inflight_entertainment                 0
On-board_service                       0
Leg_room_service                       0
Baggage_handling                       0
Checkin_service                        0
Inflight_service                       0
Cleanliness                            0
Departure_Delay_in_Minutes             0
Arrival_Delay_in_Minutes             310
satisfaction                           0
dtype: int64
```

```python
# Null Values After fillna
display(df_train.isnull().sum())
display(df_test.isnull().sum())
```

```
Gender                                 0
Customer_Type                          0
Age                                    0
Type_of_Travel                         0
Class                                  0
Flight_Distance                        0
Inflight_wifi_service                  0
Departure/Arrival_time_convenient      0
Ease_of_Online_booking                 0
Gate_location                          0
Food_and_drink                         0
Online_boarding                        0
Seat_comfort                           0
Inflight_entertainment                 0
On-board_service                       0
Leg_room_service                       0
Baggage_handling                       0
Checkin_service                        0
Inflight_service                       0
Cleanliness                            0
Departure_Delay_in_Minutes             0
Arrival_Delay_in_Minutes               0
satisfaction                           0
dtype: int64
```

```python
# Set the null values of Arrival Delay to Departure Delay since if one departs a certain time late, they will most likely a
df_train['Arrival_Delay_in_Minutes'] = df_train['Arrival_Delay_in_Minutes'].fillna(df_train['Departure_Delay_in_Minutes'])
df_test['Arrival_Delay_in_Minutes'] = df_test['Arrival_Delay_in_Minutes'].fillna(df_test['Departure_Delay_in_Minutes'])
```

```
# Data Standardization

scaler = StandardScaler()
numCols = df_train.select_dtypes(include=np.number).columns
df_train[numCols] = pd.DataFrame(scaler.fit_transform(df_train[numCols]), columns=df_train[numCols].columns)
display(df_train)

df_test[numCols] = pd.DataFrame(scaler.fit_transform(df_test[numCols]), columns=df_test[numCols].columns)
```

| . | Inflight_Entertainment | On-board_Service | Leg_Room_Service | Baggage_Handling | Checkin_Service | Inflight_Service | Cleanliness | Departure_Delay_in_Minutes | Arrival_Delay_in_Minutes |
|---|---|---|---|---|---|---|---|---|---|
| . | 1.231704 | 0.479403 | -0.266840 | 0.311769 | 0.549799 | 1.156436 | 1.305870 | 0.266393 | 0.070988 |
| . | -1.769081 | -1.849161 | 1.253380 | -0.535045 | -1.821012 | 0.305848 | -1.742292 | -0.361375 | -0.238223 |
| . | 1.231704 | 0.479403 | -0.266840 | 0.311769 | 0.549799 | 0.305848 | 1.305870 | -0.387532 | -0.392828 |
| . | -1.018885 | -1.072973 | 1.253380 | -0.535045 | -1.821012 | 0.305848 | -0.980251 | -0.099805 | -0.160920 |
| . | -0.268688 | -0.296785 | 0.493270 | 0.311769 | -0.240472 | -0.544740 | -0.218211 | -0.387532 | -0.392828 |
| . | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| . | -1.018885 | -0.296785 | -1.787061 | 0.311769 | -1.030742 | -0.544740 | -0.980251 | -0.309061 | -0.392828 |
| . | 1.231704 | 1.255590 | 1.253380 | 1.158582 | 1.340069 | 1.156436 | 0.543829 | -0.387532 | -0.392828 |
| . | 0.481508 | -0.296785 | -1.026951 | 0.311769 | 1.340069 | 1.156436 | 0.543829 | -0.204433 | -0.032082 |
| . | -1.769081 | 0.479403 | 1.253380 | -2.228672 | 1.340069 | 0.305848 | -1.742292 | -0.387532 | -0.392828 |
| . | -1.769081 | -1.849161 | -1.787061 | 0.311769 | 0.549799 | -0.544740 | -1.742292 | -0.387532 | -0.392828 |

```
# All the different values for the Categorical Columns

print(df_train['Gender'].unique())
print(df_train['Customer_Type'].unique())
print(df_train['Type_of_Travel'].unique())
print(df_train['Class'].unique())
```

```
['Male' 'Female']
['Loyal_Customer' 'Disloyal_Customer']
['Personal Travel' 'Business_Travel']
['Eco Plus' 'Business' 'Eco']
```

```
# One Hot Encoding for Categorical Features

categorical_columns = ['Gender', 'Customer_Type', 'Type_of_Travel']
ohe = OneHotEncoder(sparse_output=False)
one_hot_encoded = ohe.fit_transform(df_train[categorical_columns])
one_hot_df = pd.DataFrame(one_hot_encoded, columns=ohe.get_feature_names_out(categorical_columns))

encoded_train = pd.concat([one_hot_df, df_train], axis=1)
encoded_train = encoded_train.drop(categorical_columns, axis=1)
display(encoded_train)

categorical_columns = ['Gender', 'Customer_Type', 'Type_of_Travel']
ohe = OneHotEncoder(sparse_output=False)
one_hot_encoded = ohe.fit_transform(df_test[categorical_columns])
one_hot_df = pd.DataFrame(one_hot_encoded, columns=ohe.get_feature_names_out(categorical_columns))

encoded_test = pd.concat([one_hot_df, df_test], axis=1)
encoded_test = encoded_test.drop(categorical_columns, axis=1)
```

| Gender_Female | Gender_Male | Customer_Type_Disloyal_Customer | Customer_Type_Loyal_Customer | Type_of_Travel_Business_Travel | Type_of_Travel_Personal Travel |
|---|---|---|---|---|---|
| 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 |
| 0.0 | 1.0 | 1.0 | 0.0 | 1.0 | 0.0 |
| 1.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 |
| 1.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 |
| 0.0 | 1.0 | 0.0 | 1.0 | 1.0 | 0.0 |
| ... | ... | ... | ... | ... | ... |
| 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 |
| 0.0 | 1.0 | 0.0 | 1.0 | 1.0 | 0.0 |
| 0.0 | 1.0 | 1.0 | 0.0 | 1.0 | 0.0 |
| 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 |
| 0.0 | 1.0 | 0.0 | 1.0 | 1.0 | 0.0 |

```
# Ordinal Encoding for the Class where the order is Eco, Eco Plus, and then Business

enc = OrdinalEncoder(categories=[['Eco', 'Eco Plus','Business']])
encoded_train['Class'] = enc.fit_transform(encoded_train.loc[:,['Class']])
enc = OrdinalEncoder(categories=[['Neutral_Or_Dissatisfied', 'Satisfied']])
encoded_train['Satisfaction'] = enc.fit_transform(encoded_train.loc[:,['Satisfaction']])

display(encoded_train)

enc = OrdinalEncoder(categories=[['Eco', 'Eco Plus','Business']])
encoded_test['Class'] = enc.fit_transform(encoded_test.loc[:,['Class']])
enc = OrdinalEncoder(categories=[['Neutral_Or_Dissatisfied', 'Satisfied']])
encoded_test['Satisfaction'] = enc.fit_transform(encoded_test.loc[:,['Satisfaction']])
```

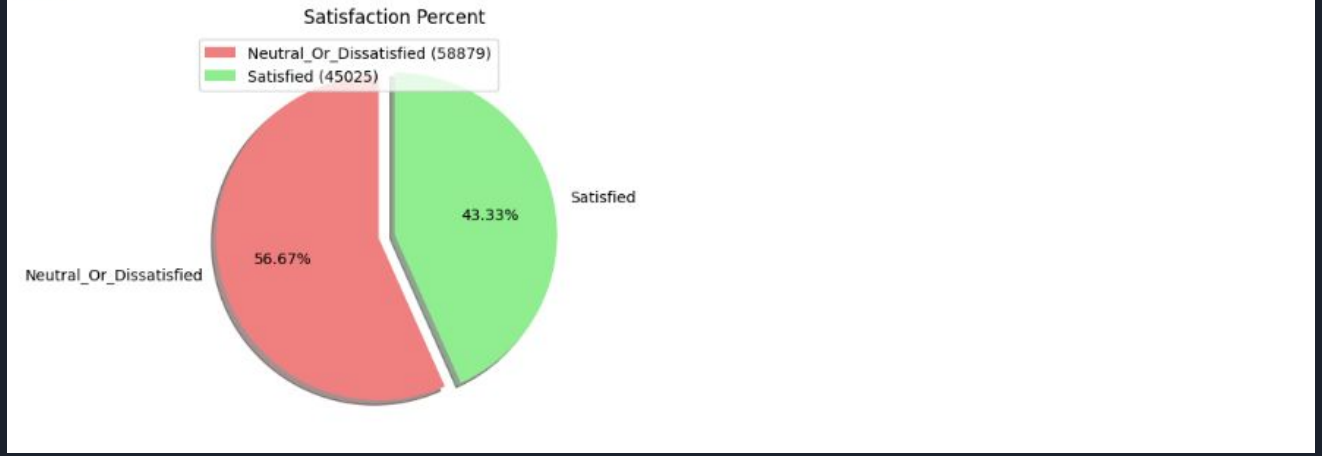
```
# Standardized Encoded Data
std_encoded_train = encoded_train
std_encoded_train['Class'] = pd.Series(scaler.fit_transform(encoded_train[['Class']]).flatten())
display(std_encoded_train)

std_encoded_test = encoded_test
std_encoded_test['Class'] = pd.Series(scaler.fit_transform(encoded_test[['Class']]).flatten())
```

| Class | Satisfaction |
|---|---|
| 1.0 | 0.0 |
| 2.0 | 0.0 |
| 2.0 | 1.0 |
| 2.0 | 0.0 |
| 2.0 | 1.0 |
| ... | ... |
| 0.0 | 0.0 |
| 2.0 | 1.0 |
| 2.0 | 0.0 |
| 0.0 | 0.0 |
| 2.0 | 0.0 |

# Exploratory Data Analysis

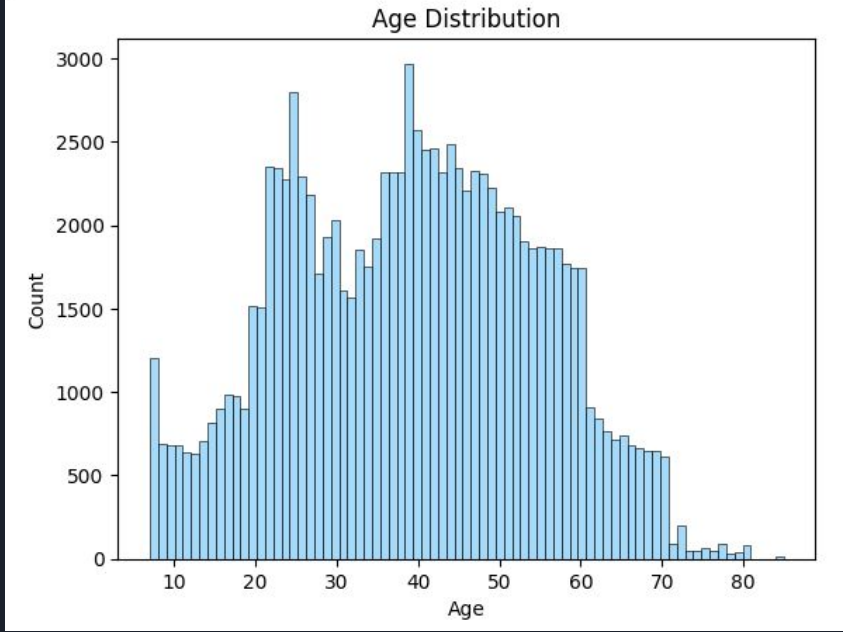
# Satisfaction Percentage

```
fig, axes = plt.subplots(nrows=1, ncols=1)
axes.pie(df_train['Satisfaction'].value_counts(),
         labels=df_train['Satisfaction'].value_counts().index,
         explode=[0.1, 0],
         autopct='%1.2f%%',
         shadow=True,
         startangle=90,
         colors=["lightcoral", "lightgreen"])
axes.set_title('Satisfaction Percent')
legend_labels = [f'{label} ({count})' for label, count in zip(df_train['Satisfaction'].value_counts().index, df_train['Satisfaction'].value_counts())]
axes.legend(legend_labels, loc='upper left')
plt.show()
```



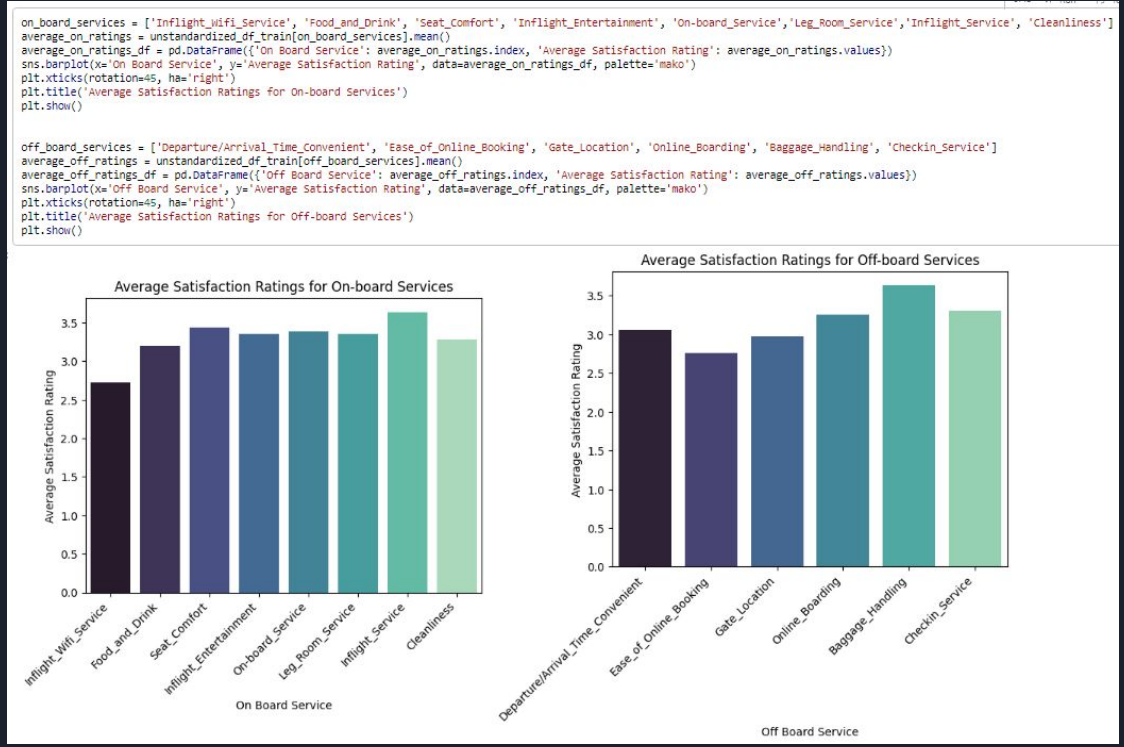
# Age Distribution

```
sns.histplot(data=unstandardized_df_train,x='Age', color="lightskyblue")
plt.title("Age Distribution")
plt.show()
```



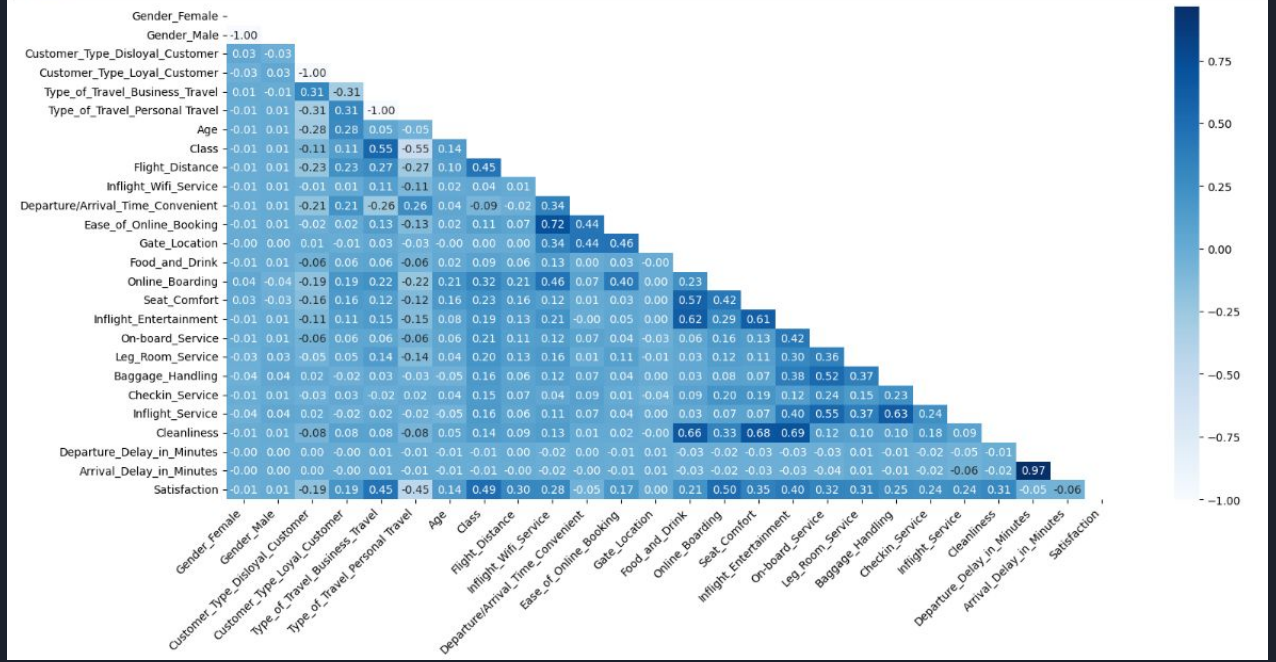
# Average Satisfaction on Services

```
on_board_services = ['Inflight_Wifi_Service', 'Food_and_Drink', 'Seat_Comfort', 'Inflight_Entertainment', 'On-board_Service','Leg_Room_Service','Inflight_Service', 'Cleanliness']
average_on_ratings = unstandardized_df_train[on_board_services].mean()
average_on_ratings_df = pd.DataFrame({'On Board Service': average_on_ratings.index, 'Average Satisfaction Rating': average_on_ratings.values})
sns.barplot(x='On Board Service', y='Average Satisfaction Rating', data=average_on_ratings_df, palette='mako')
plt.xticks(rotation=45, ha='right')
plt.title('Average Satisfaction Ratings for On-board Services')
plt.show()


off_board_services = ['Departure/Arrival_Time_Convenient', 'Ease_of_Online_Booking', 'Gate_Location', 'Online_Boarding', 'Baggage_Handling', 'Checkin_Service']
average_off_ratings = unstandardized_df_train[off_board_services].mean()
average_off_ratings_df = pd.DataFrame({'Off Board Service': average_off_ratings.index, 'Average Satisfaction Rating': average_off_ratings.values})
sns.barplot(x='Off Board Service', y='Average Satisfaction Rating', data=average_off_ratings_df, palette='mako')
plt.xticks(rotation=45, ha='right')
plt.title('Average Satisfaction Ratings for Off-board Services')
plt.show()
```



# Correlation Matrix

```
plt.figure(figsize=(18,8))

corr = std_encoded_train.corr()
mask = np.triu(corr)

sns.heatmap(corr, annot=True, fmt='.2f', mask=mask, cmap="Blues")
plt.xticks(rotation=45, ha='right')
plt.show()
```

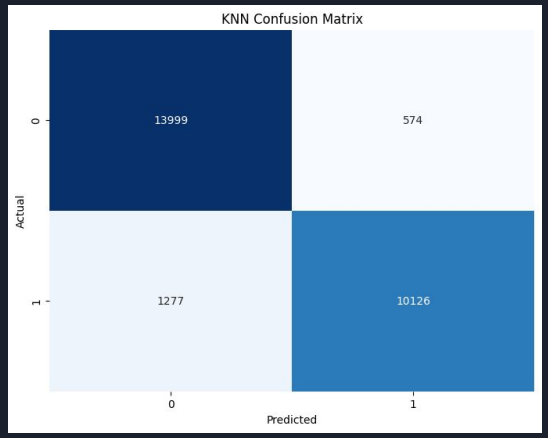


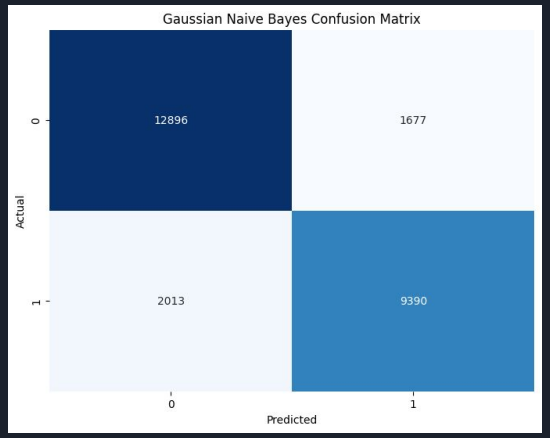
# Model Building Techniques Used

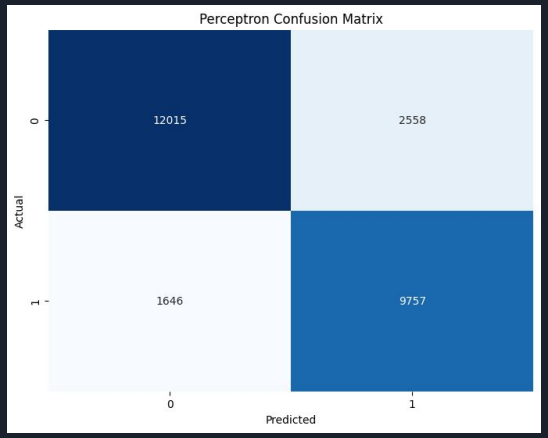
- K-Nearest Neighbors
- Gaussian Naive Bayes
- Perceptron
- Logistic Regression
- Decision Trees
- Random Forest Classifier

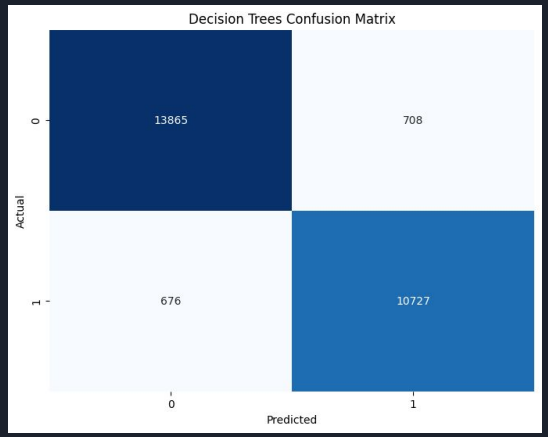
```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
classifiers = [("KNN", KNeighborsClassifier()),
               ("Gaussian Naive Bayes", GaussianNB()),
               ("Perceptron", Perceptron()),
               ("Logistic Regression", LogisticRegression()),
               ("Decision Trees", DecisionTreeClassifier()),
               ("Random Forest Classifier", RandomForestClassifier())]

for name, classifier in classifiers:
    classifier.fit(X_train, Y_train)
```

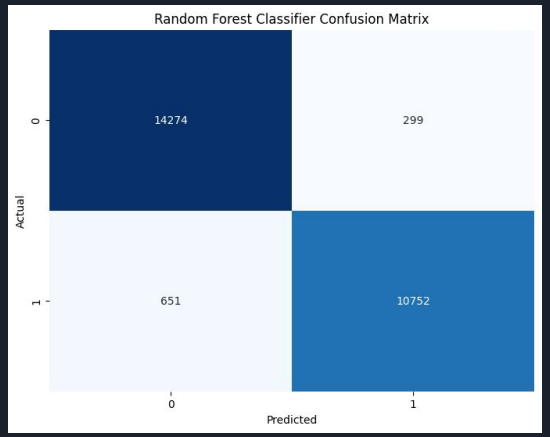
```python
from sklearn.metrics import accuracy_score, confusion_matrix, recall_score, precision_score, f1_score

results = pd.DataFrame(columns=["Classifier", "Accuracy", "Recall", "Precision", "f1"])

for name, classifier in classifiers:
    Y_pred = classifier.predict(X_test)
    accuracy = accuracy_score(Y_test, Y_pred)
    recall = recall_score(Y_test, Y_pred)
    precision = precision_score(Y_test, Y_pred)
    f1 = f1_score(Y_test, Y_pred)
    print("Name: ", name)
    data = pd.DataFrame([{"Classifier": name, "Accuracy": accuracy, "Recall": recall, "Precision": precision, "f1": f1}])
    confusion = confusion_matrix(Y_test, Y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues", cbar=False)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(name + ' Confusion Matrix')
    plt.show()
    results = pd.concat([results, data], ignore_index=True)
```

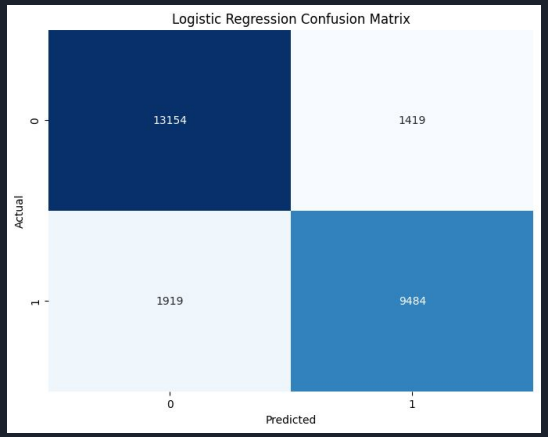

KNN Confusion Matrix
13999
574
1277
10126
Actual
Predicted
0
1


Gaussian Naive Bayes Confusion Matrix
12896
1677
2013
9390
Actual
Predicted
0
1


Perceptron Confusion Matrix
12015
2558
1646
9757
Actual
Predicted
0
1


Decision Trees Confusion Matrix
13865
708
676
10727
Actual
Predicted
0
1


Random Forest Classifier Confusion Matrix
14274
299
651
10752
Actual
Predicted
0
1


Logistic Regression Confusion Matrix
13154
1419
1919
9484
Actual
Predicted
0
1

# Model Comparison

```
styled_results = results.style.background_gradient(cmap='Blues', subset=["Accuracy", "Recall", "Precision", "f1"])
display(styled_results)
```

| | Classifier | Accuracy | Recall | Precision | f1 |
|---|---|---|---|---|---|
| 0 | KNN | 0.928742 | 0.888012 | 0.946355 | 0.916256 |
| 1 | Gaussian Naive Bayes | 0.857946 | 0.823468 | 0.848468 | 0.835781 |
| 2 | Perceptron | 0.838158 | 0.855652 | 0.792286 | 0.822751 |
| 3 | Logistic Regression | 0.871497 | 0.831711 | 0.869852 | 0.850354 |
| 4 | Decision Trees | 0.947298 | 0.940805 | 0.939240 | 0.940022 |
| 5 | Random Forest Classifier | 0.964044 | 0.942559 | 0.974698 | 0.958359 |