Part 3 Model Evaluation and Comparison

In Part 3, we used predictive modeling aspect of our churn prediction project and to evaluate various machine learning models to determine which performs best at predicting customer churn based on the dataset provided.

The dataset we used was Churn_Modelling.csv

We used LabelEncoder to numerically transform the Geography and Gender columns, converting categorical information into a format understandable by our models. Columns like RowNumber, CustomerId, and Surname were removed since they contributed no predictive value to the churn outcome. To standardize the range of our continuous input features, we employed StandardScaler. This step is particularly beneficial for distance-based models and can significantly impact their performance.

Machine Learning Models: Naïve Bayes (GaussianNB), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree, and Logistic Regression. Each model was trained on the preprocessed training dataset and evaluated on a separate testing dataset to gauge its predictive accuracy.

When we created the model For KNN, the K value for K (neighbors) and that gave us the highest accuracy was 20.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load the dataset
df = pd.read_csv('Churn_Modelling.csv')

# Encode categorical variables
label_encoder = LabelEncoder()
df['Geography'] = label_encoder.fit_transform(df['Geography'])
df['Gender'] = label_encoder.fit_transform(df['Gender'])

# Dropping irrelevant features
df = df.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1)

# Splitting the dataset
X = df.drop('Exited', axis=1)
y = df['Exited']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Training and evaluating models
models = {
    'Naïve Bayes': GaussianNB(),
    'KNN': KNeighborsClassifier(n_neighbors=20),
    'SVM': SVC(),
    'Decision Tree': DecisionTreeClassifier(),
    'Logistic Regression': LogisticRegression(max_iter=1000)
}

accuracies = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    accuracies[name] = accuracy_score(y_test, predictions)

# Displaying accuracies
for model, acc in accuracies.items():
    print(f"{model}: {acc * 100:.2f}%")
```

```
Naïve Bayes: 82.85%
KNN (K-Value = 20): 84.25%
SVM: 85.75%
Decision Tree: 77.70%
Logistic Regression: 81.50%
```