LOGISTIC

DELIVERY DELAY PREDICTION

MACHINE LEARNING

PROJECT BY - JITHIN JAYACHANDRAN

INTRODUCTION

PROJECT OBJECTIVE
THE OBJECTIVE OF THIS PROJECT IS TO DEVELOP A MACHINE LEARNING MODEL THAT CAN ACCURATELY PREDICT DELIVERY OUTCOMES IN A REAL-WORLD LOGISTICS SUPPLY CHAIN SYSTEM. SPECIFICALLY, THE GOAL IS TO CLASSIFY DELIVERIES INTO THREE CATEGORIES:

-1 (EARLY): THE DELIVERY ARRIVES BEFORE THE EXPECTED TIME.
0 (ON-TIME): THE DELIVERY ARRIVES EXACTLY AS SCHEDULED.
1 (DELAYED): THE DELIVERY ARRIVES AFTER THE SCHEDULED TIME.
BY LEVERAGING VARIOUS FEATURES RELATED TO THE LOGISTICS PROCESS (SUCH AS ORDER AND SHIPPING DATES, PAYMENT TYPES, AND SALES PER CUSTOMER), THE PROJECT AIMS TO BUILD A MODEL THAT ASSISTS BUSINESSES IN OPTIMIZING THEIR SUPPLY CHAIN OPERATIONS, IMPROVING DELIVERY ACCURACY, AND ENHANCING CUSTOMER SATISFACTION.

PROBLEM STATEMENT:
MULTI-LABEL CLASSIFICATION OF DELIVERIES WITH CLASS IMBALANCE.
ACCURATE PREDICTION OF DELIVERY STATUS TO OPTIMIZE SUPPLY CHAIN EFFICIENCY AND CUSTOMER SATISFACTION.

DATASET OVERVIEW:
FEATURES: PAYMENT TYPE, PROFIT, SALES PER CUSTOMER, CATEGORY ID, CUSTOMER CITY, ORDER/SHIPPING DATES, ETC.
TARGET: DELIVERY STATUS (LABEL WITH VALUES -1, 0, 1).
CHALLENGES: HANDLING CLASS IMBALANCE, CATEGORICAL/NUMERICAL FEATURE PREPROCESSING.

ASSOCIATED TASKS:
DATA PREPROCESSING: HANDLE MISSING VALUES, CLASS IMBALANCE, AND ENCODE FEATURES.
EDA: VISUALIZE FEATURE RELATIONSHIPS AND IDENTIFY OUTLIERS.
MODEL TRAINING: TEST MULTIPLE MODELS (LOGISTIC REGRESSION, DECISION TREES, RANDOM FOREST, SVM, KNN).
MODEL EVALUATION: ASSESS ACCURACY, PRECISION, RECALL, F1-SCORE.
OPTIMIZATION: FINE-TUNE HYPERPARAMETERS.

IMPORTANCE:
CUSTOMER SATISFACTION: MORE RELIABLE DELIVERY TIMELINES.
COST REDUCTION: OPTIMIZE STORAGE AND LABOR BASED ON PREDICTED DELIVERY STATUS.
OPERATIONAL EFFICIENCY: BETTER RESOURCE ALLOCATION.
COMPETITIVE ADVANTAGE: ENHANCED SERVICE AND BRAND REPUTATION.

DATASET OVERVIEW

DATASET NAME: LOGISTICS SUPPLY CHAIN REAL WORLD DATA

TOTAL INSTANCES: 100,000+ ROWS
TARGET VARIABLE: LABEL

VALUES:
-1: EARLY DELIVERY
0: ON-TIME DELIVERY
1: DELAYED DELIVERY

FEATURES:
PAYMENT INFORMATION:
PAYMENT_TYPE: (CREDIT CARD, COD, ETC.)
PROFITABILITY & SALES:
PROFIT_PER_ORDER: PROFIT MARGIN FOR EACH ORDER
SALES_PER_CUSTOMER: TOTAL SALES PER CUSTOMER
ORDER & SHIPPING DETAILS:
ORDER_DATE: DATE THE ORDER WAS PLACED
SHIPPING_DATE: DATE THE ORDER WAS SHIPPED
ORDER_STATUS: (SHIPPED, DELIVERED, CANCELED, ETC.)
CUSTOMER & PRODUCT INFO:
CUSTOMER_CITY: LOCATION OF THE CUSTOMER
CATEGORY_ID: PRODUCT CATEGORY ORDERED
OTHER METADATA:
LEAD TIME, SHIPPING MODE: OTHER FEATURES INFLUENCING DELIVERY SPEED
DATASET CHARACTERISTICS:

NO MISSING VALUES ACROSS CRITICAL FEATURES.
CLASS IMBALANCE: MORE ON-TIME DELIVERIES, FEWER EARLY OR DELAYED.
MIXED DATA TYPES: CATEGORICAL (PAYMENT_TYPE, ORDER_STATUS), NUMERICAL (PROFIT, SALES, LEAD TIME).

A	B	C	D	E	F	G	H	I	J	K
payment_type	profit_per_order	sales_per_customer	category_id	category_name	customer_city	customer_country	customer_id	customer_segment	customer_state	customer_zipcode
DEBIT	34.448338	92.49099	9	Cardio Equipment	Caguas	Puerto Rico	12097.683	Consumer	PR	725
TRANSFER	91.19354	181.99008	48	Water Sports	Albuquerque	EE. UU.	5108.1045	Consumer	CA	92745.16
DEBIT	8.313806	89.96643	46	ndoor/Outdoor Game	Amarillo	Puerto Rico	4293.4478	Consumer	PR	2457.7297
TRANSFER	-89.463196	99.15065	17	Cleats	Caguas	Puerto Rico	546.5306	Consumer	PR	725
DEBIT	44.72259	170.97824	48	Water Sports	Peabody	EE. UU.	1546.398	Consumer	CA	95118.6
CASH	76.1004	137.4536	17	Electronics	Caguas	Puerto Rico	5048.3975	Consumer	PR	725
DEBIT	-54.34529	167.98117	46	ndoor/Outdoor Game	Caguas	Puerto Rico	7413.2383	Corporate	PR	725
TRANSFER	-163.6284	120.89	18	Men's Footwear	Caguas	Puerto Rico	6775.2695	Home Office	PR	725
DEBIT	29.792816	113.09	18	Men's Footwear	Hanford	EE. UU.	4784.4346	Consumer	KY	28629.11

L	M	N	O	P	Q	R	S	T	U	V
department_id	department_name	latitude	longitude	market	order_city	order_country	order_customer_id	order_date	order_id	rder_item_cardprod_
3	Footwear	18.359064	-66.370575	Europe	Viena	Austria	12073.336	15-08-12 00:00:00+01	15081.289	191
7	Fan Shop	37.636528	-121.11963	LATAM	Buenos Aires	Argentina	5111.048	17-02-10 00:00:00+00	56444.684	1073
7	Fan Shop	18.2941	-66.037056	Europe	Burnie	France	4134.765	15-01-01 00:00:00+00	7508.5713	1014
4	Apparel	18.202435	-66.37051	LATAM	Santa Ana	El Salvador	495.18726	17-05-31 00:00:00+01	56196.926	365
7	Fan Shop	38.7195	-122.31972	LATAM	Blumenau	Mexico	1758.9119	15-03-28 00:00:00+00	5565.5796	1073
3	Footwear	18.26879	-66.3705	USCA	Philadelphia	United States	4891.248	16-06-06 00:00:00+01	32955.824	365
7	Fan Shop	18.244066	-66.37058	USCA	Los Angeles	United States	7524.8945	16-05-17 00:00:00+01	35385.855	1014
4	Apparel	18.214336	-66.37052	USCA	Philadelphia	United States	6665.7886	16-06-09 00:00:00+01	36338.4	403
4	Apparel	38.620014	-84.38348	Pacific Asia	Bangkok	Thailand	4701.6694	16-06-06 00:00:00+01	27692.854	403
5	Golf	18.264816	-66.37061	LATAM	Villahermosa	Mexico	11918.106	17-08-29 00:00:00+01	56918.418	627

W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG
order_item_discount	der_item_discount_ra	order_item_id	der_item_product_pri	rder_item_profit_rati	order_item_quantity	sales	der_item_total_amou	rder_profit_per_orde	order_region	order_state
12.623338	0.13	38030.996	99.99	0.41	1	99.99	84.99157	32.083145	Western Europe	Vienna
16.5	0.07	142621.78	199.99	0.48	1	199.99	181.99	91.23587	South America	Buenos Aires
6.6	0.06	18723.178	49.98	0.09	2	99.96	93.81015	6.9655495	Western Europe	rd-Pas-de-Calais-Pica
16.942171	0.16	141654.58	59.99	-0.8	2	119.98	99.8906	-95.4014	Central America	Santa Ana
29.99	0.15	14204.896	199.99	0.27	1	199.99	171.07587	44.569	Central America	Illinois
0	0	79153.03	39.99	0.49	4	129.99	145.46329	81.08791	East of USA	Pennsylvania
35.99	0.2	89402.43	49.98	-0.56984335	4	199.92	167.99	-72.914665	West of USA	California
15.6	0.1	87497.61	129.99	-1.55	1	129.99	116.99	-169.01434	East of USA	Pennsylvania
18	0.13	67108.8	129.99	0.27	1	129.99	113.15623	27.000895	Southeast Asia	Bangkok
31.404892	0.18	141197.36	39.99	0.099622846	4	159.96	127.39	11.838907	Central America	Tabasco

AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO
rder_profit_per_orde	order_region	order_state	order_status	product_card_id	product_category_id	product_name	product_price	shipping_date	shipping_mode	label
32.083145	Western Europe	Vienna	COMPLETE	191	9	len's Free 5.0+ Runnin	99.99	15-08-13 00:00:00+01	Standard Class	-1
91.23587	South America	Buenos Aires	PENDING	1073	48	can Sunstream 100 Ka	199.99	17-04-09 00:00:00+01	Standard Class	-1
6.9655495	Western Europe	rd-Pas-de-Calais-Pica	COMPLETE	1014	46	n Men's Neoprene Lif	49.98	15-03-18 00:00:00+00	Second Class	1
-95.4014	Central America	Santa Ana	PROCESSING	365	17	ct Fitness Perfect Rip	59.99	17-03-18 00:00:00+00	Second Class	0
44.569	Central America	Illinois	COMPLETE	1073	48	can Sunstream 100 Ka	199.99	15-03-30 00:00:00+01	Standard Class	1
81.08791	East of USA	Pennsylvania	CLOSED	209.90128	9	hour Women's Ignite F	39.99	16-10-14 00:00:00+01	Standard Class	1
-72.914665	West of USA	California	COMPLETE	1014	46	n Men's Neoprene Lif	49.98	16-07-03 00:00:00+01	Standard Class	1
-169.01434	East of USA	Pennsylvania	PROCESSING	403	18	n's CJ Elite 2 TD Foott	129.99	16-04-24 00:00:00+01	Standard Class	-1
27.000895	Southeast Asia	Bangkok	ON_HOLD	403	18	n's CJ Elite 2 TD Foott	129.99	16-12-23 00:00:00+00	Standard Class	1
11.838907	Central America	Tabasco	PENDING_PAYMENT	627	29	ur Girls' Toddler Spine	39.99	17-04-28 00:00:00+01	First Class	1

LOAD THE DATASET

```
[1] import pandas as pd

    # Load the dataset
    df = pd.read_csv('/content/incom2024_delay_example_dataset.csv')
    df.head()
```

	payment_type	profit_per_order	sales_per_customer	category_id	category_name	customer_city	customer_country	customer_id	customer_segment
0	DEBIT	34.448338	92.49099	9.0	Cardio Equipment	Caguas	Puerto Rico	12097.6830	Consumer
1	TRANSFER	91.193540	181.99008	48.0	Water Sports	Albuquerque	EE. UU.	5108.1045	Consumer
2	DEBIT	8.313806	89.96643	46.0	Indoor/Outdoor Games	Amarillo	Puerto Rico	4293.4478	Consumer
3	TRANSFER	-89.463196	99.15065	17.0	Cleats	Caguas	Puerto Rico	546.5306	Consumer
4	DEBIT	44.722590	170.97824	48.0	Water Sports	Peabody	EE. UU.	1546.3980	Consumer

EDA & PRE-PROCESSING

```
[2] # Checking for missing values
    missing_values = df.isnull().sum()
    print("Missing Values:\n", missing_values)
```

```
Missing Values:
 payment_type              0
profit_per_order          0
sales_per_customer        0
category_id               0
category_name             0
customer_city             0
customer_country          0
customer_id               0
customer_segment          0
customer_state            0
customer_zipcode          0
department_id             0
department_name           0
latitude                  0
longitude                 0
market                    0
order_city                0
order_country             0
order_customer_id         0
order_date                0
order_id                  0
order_item_cardprod_id    0
order_item_discount       0
order_item_discount_rate  0
order_item_id             0
order_item_product_price  0
order_item_profit_ratio   0
order_item_quantity       0
sales                     0
order_item_total_amount   0
order_profit_per_order    0
order_region              0
order_state               0
order_status              0
product_card_id           0
product_category_id       0
product_name              0
```

```
[3] # Analyzing categorical columns
    categorical_columns = df.select_dtypes(include=['object']).nunique()
    print("\nCategorical Columns:\n", categorical_columns)
```

```
Categorical Columns:
 payment_type          4
category_name        49
customer_city       555
customer_country      2
customer_segment      3
customer_state       44
department_name      11
market                5
order_city         2742
order_country       152
order_date         1162
order_region         23
order_state         982
order_status          7
product_name        113
shipping_date      1170
shipping_mode         4
dtype: int64
```

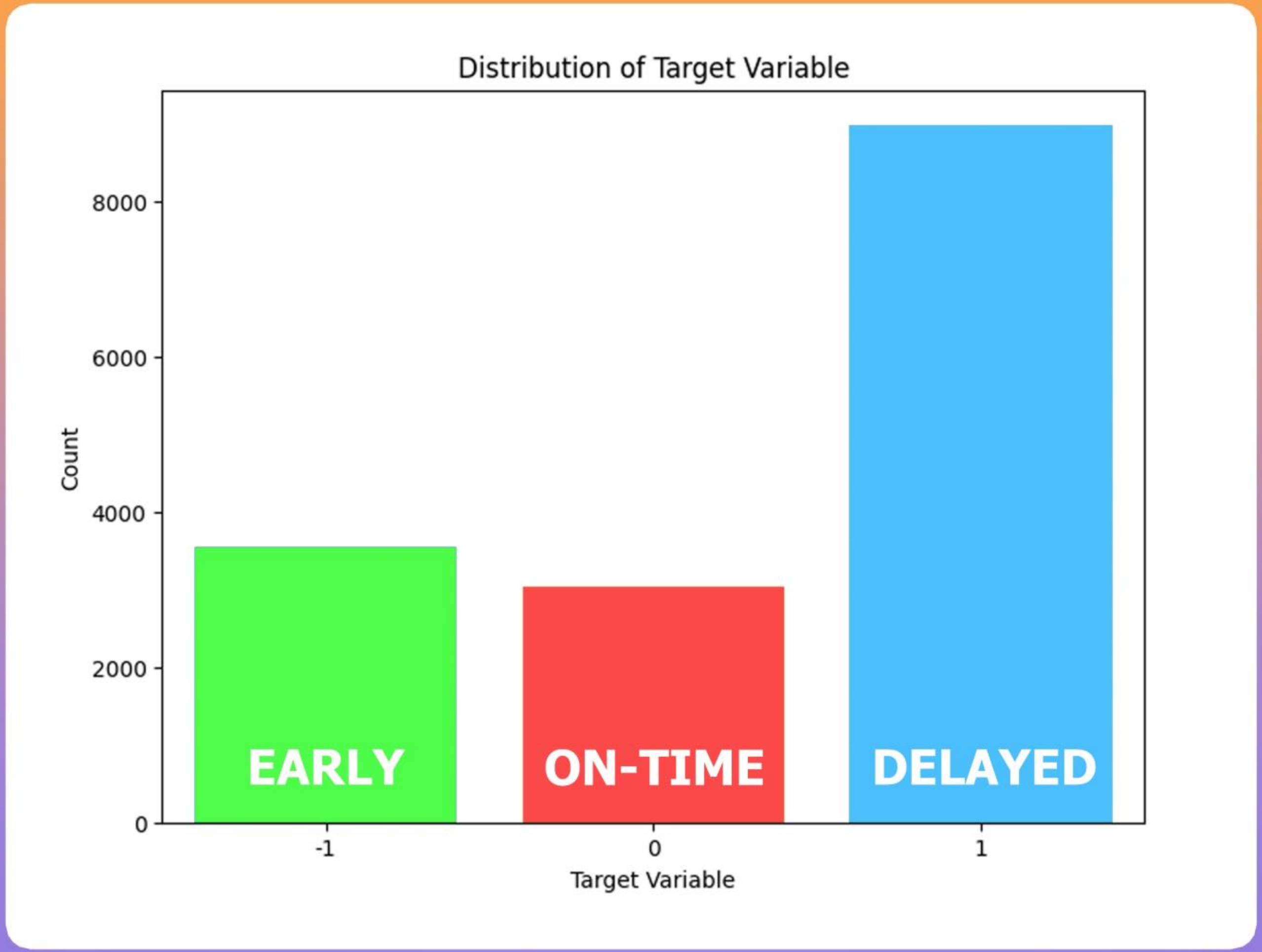
```
[5] # Analyzing numerical columns
    numerical_columns = df.describe()
    print("\nNumerical Columns:\n", numerical_columns)
```

```
std          99.265198        113.727323    15.303616   4114.273782
min       -3442.500000          8.351162     2.000000      1.000000
25%           7.562795        104.397330    18.000000   3119.983200
50%          31.693370        165.944170    29.000000   6429.229000
75%          63.872166        242.440930    45.000000   9642.381000
max         911.800000       1939.990000    76.000000  20757.000000

       customer_zipcode  department_id      latitude     longitude  \
count      15549.000000   15549.000000  15549.000000  15549.000000
mean       35458.234968       5.413462     29.677619    -84.512318
std        37343.702033       1.581550      9.877876     20.681015
min          603.000000       2.000000    -33.937553   -158.025990
25%          725.000000       4.000000     18.263327    -98.088170
50%        19145.775000       5.000000     33.435677    -76.580800
75%        77502.820000       7.000000     39.277313    -66.370575
max        99205.000000      12.000000     48.781933    115.263080
```

```
[4] df.dtypes
```

	0
payment_type	object
profit_per_order	float64
sales_per_customer	float64
category_id	float64
category_name	object
customer_city	object
customer_country	object
customer_id	float64
customer_segment	object
customer_state	object
customer_zipcode	float64
department_id	float64
department_name	object
latitude	float64
longitude	float64
market	object
order_city	object
order_country	object
order_customer_id	float64
order_date	object

DISTRIBUTION OF TARGET VARIABLES



CLASS BALANCING

```python
# Ensure 'label' is of integer type
df['label'] = df['label'].astype(int)

# Check the class distribution
print("Class Distribution before balancing:\n", df['label'].value_counts())

# Handling the class imbalance by removing excess rows of class 1
class_counts = df['label'].value_counts()
min_class_count = class_counts.min()  # This will be the number of rows to keep for all classes

# Randomly sample 'min_class_count' rows from class 1
df_class1 = df[df['label'] == 1].sample(n=min_class_count, random_state=42)
df_class_minus1 = df[df['label'] == -1]
df_class0 = df[df['label'] == 0]

# Combine the balanced dataset
df_balanced = pd.concat([df_class1, df_class_minus1, df_class0], axis=0)

# Shuffle the dataset
df_balanced = df_balanced.sample(frac=1, random_state=42).reset_index(drop=True)

# Check the new class distribution
print("Class Distribution after balancing:\n", df_balanced['label'].value_counts())
```

```
Class Distribution before balancing:
 label
 1    8976
-1    3545
 0    3028
Name: count, dtype: int64
Class Distribution after balancing:
 label
-1    3545
 1    3028
 0    3028
Name: count, dtype: int64
```

HANDLING OUTLIERS

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Identify numerical columns
numerical_columns = df_balanced.select_dtypes(include=['float64', 'int64']).columns

# Visualize outliers using boxplots
for column in numerical_columns:
    plt.figure(figsize=(10, 5))
    sns.boxplot(x=df_balanced[column])
    plt.title(f'Boxplot of {column}')
    plt.show()
```

```python
import numpy as np

# Handling outliers by capping them to a threshold based on IQR
for column in numerical_columns:
    Q1 = df_balanced[column].quantile(0.25)
    Q3 = df_balanced[column].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Capping the outliers
    df_balanced[column] = np.where(df_balanced[column] < lower_bound, lower_bound, df_balanced[column])
    df_balanced[column] = np.where(df_balanced[column] > upper_bound, upper_bound, df_balanced[column])
```

SPLITTING THE DATA

```
[13] # Convert the target variable 'label' to categorical if it's not already
     df['label'] = df['label'].astype(int)
```

```
[14] # Splitting the data into features (X) and target (y)
     X = df_balanced.drop(columns=['label'])  # Drop the target column 'label'
     y = df_balanced['label']
```

```
[15] from sklearn.model_selection import train_test_split

     # Train-test split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

ENCODING

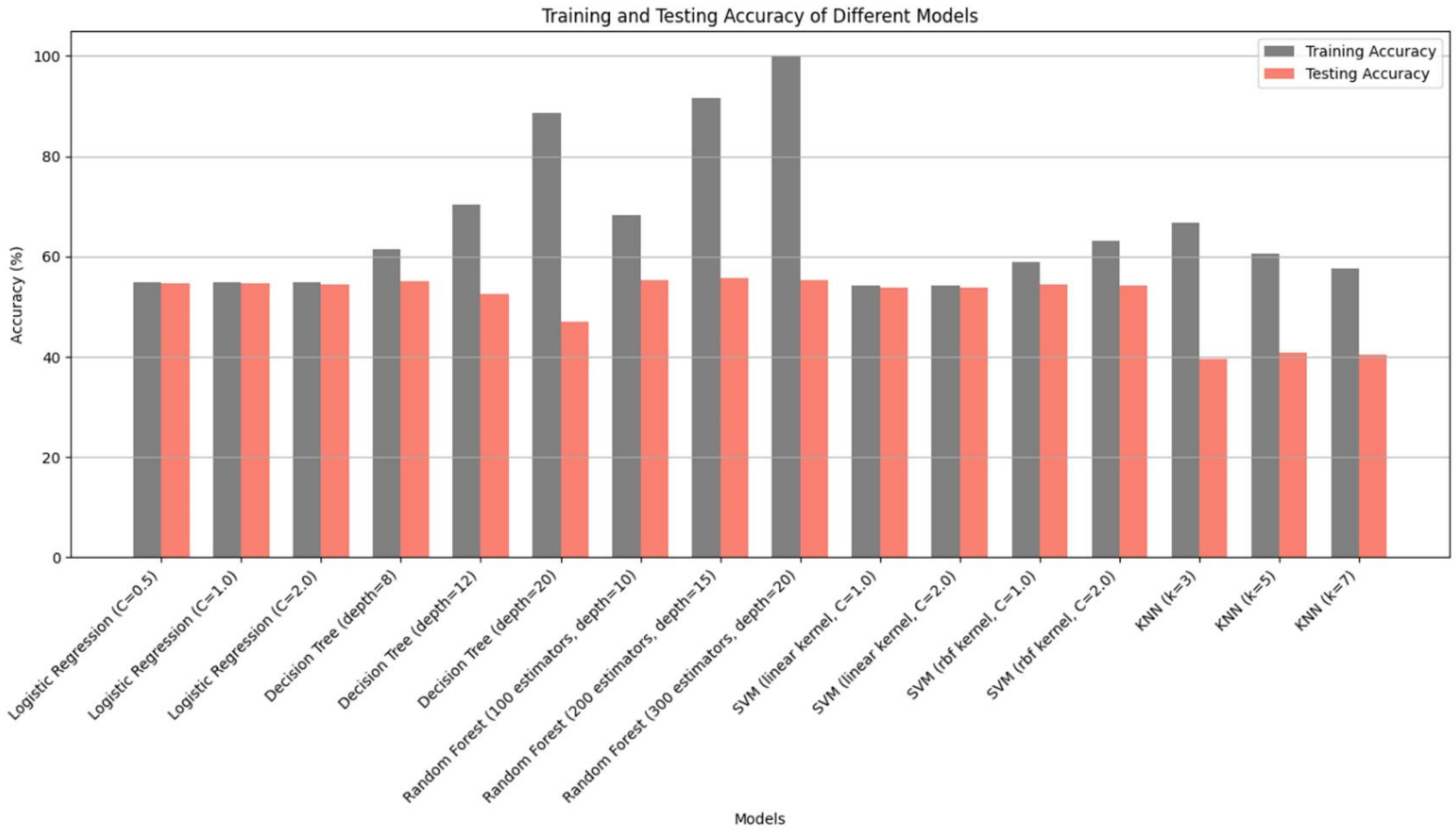
```
[36] from sklearn.preprocessing import LabelEncoder
     # Label encode categorical columns
     label_encoders = {}
     for col in df_balanced.select_dtypes(include=['object']).columns:
         le = LabelEncoder()
         df_balanced[col] = le.fit_transform(df_balanced[col].astype(str))
         label_encoders[col] = le
```

```
[43] from sklearn.preprocessing import StandardScaler

     # Apply StandardScaler to the numerical columns only (not to the categorical or label columns)
     scaler = StandardScaler()
     # Fit the scaler on the training data (only on numerical features)
     numerical_features = X_train.select_dtypes(include=['float64', 'int64']).columns
     X_train[numerical_features] = scaler.fit_transform(X_train[numerical_features])
     X_test[numerical_features] = scaler.transform(X_test[numerical_features])
```

MODEL BUILDING

```
[45] # Initialize models with hyperparameter tuning
     models = {
         # Logistic Regression: Tune the regularization parameter C
         "Logistic Regression (C=0.5)": LogisticRegression(max_iter=1000, C=0.5),
         "Logistic Regression (C=1.0)": LogisticRegression(max_iter=1000, C=1.0),
         "Logistic Regression (C=2.0)": LogisticRegression(max_iter=1000, C=2.0),

         # Decision Tree: Tune depth of the tree
         "Decision Tree (depth=8)": DecisionTreeClassifier(max_depth=8, random_state=42),
         "Decision Tree (depth=12)": DecisionTreeClassifier(max_depth=12, random_state=42),
         "Decision Tree (depth=20)": DecisionTreeClassifier(max_depth=20, random_state=42),

         # Random Forest: Tune the number of trees (n_estimators) and maximum depth
         "Random Forest (100 estimators, depth=10)": RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42),
         "Random Forest (200 estimators, depth=15)": RandomForestClassifier(n_estimators=200, max_depth=15, random_state=42),
         "Random Forest (300 estimators, depth=20)": RandomForestClassifier(n_estimators=300, max_depth=20, random_state=42),

         # SVM: Tune the kernel and regularization parameter C
         "SVM (linear kernel, C=1.0)": SVC(kernel='linear', C=1.0),
         "SVM (linear kernel, C=2.0)": SVC(kernel='linear', C=2.0),
         "SVM (rbf kernel, C=1.0)": SVC(kernel='rbf', C=1.0),
         "SVM (rbf kernel, C=2.0)": SVC(kernel='rbf', C=2.0),

         # KNN: Tune the number of neighbors (n_neighbors)
         "KNN (k=3)": KNeighborsClassifier(n_neighbors=3),
         "KNN (k=5)": KNeighborsClassifier(n_neighbors=5),
         "KNN (k=7)": KNeighborsClassifier(n_neighbors=7)
     }
```

PLOT OF TRAIN & TEST ACCURACIES



BEST MODEL SELECTION

```
[49] # Final evaluation and prediction using the best model
     final_test_accuracy = accuracy_score(y_test, best_model.predict(X_test)) * 100

     if final_test_accuracy >= acceptable_test_accuracy_threshold:
         print(f"\nFinal Evaluation of the Best Model ({best_model_name}):\n")
         final_predictions = best_model.predict(X_test)
         train_predictions = best_model.predict(X_train)

         print(f"Best Model Training Accuracy: {accuracy_score(y_train, train_predictions) * 100:.2f}%")
         print(f"Best Model Testing Accuracy: {accuracy_score(y_test, final_predictions) * 100:.2f}%")
         print("Classification Report:")
         print(classification_report(y_test, final_predictions, labels=[-1, 0, 1], zero_division=0))
     else:
         print("The model with the highest training accuracy does not meet the acceptable testing accuracy threshold.")
```

```
Final Evaluation of the Best Model (Random Forest (300 estimators, depth=20)):

Best Model Training Accuracy: 99.88%
Best Model Testing Accuracy: 55.39%
Classification Report:
              precision    recall  f1-score   support

          -1       0.52      0.94      0.67       709
           0       0.48      0.16      0.23       606
           1       0.69      0.50      0.58       606

    accuracy                           0.55      1921
   macro avg       0.56      0.53      0.49      1921
weighted avg       0.56      0.55      0.50      1921
```

CONCLUSION

BEST MODEL: RANDOM FOREST

NUMBER OF ESTIMATORS: 300
MAX DEPTH: 20
TRAINING ACCURACY: 99.88%
TESTING ACCURACY: 55.39%

2. CLASSIFICATION REPORT
PRECISION: MEASURES THE ACCURACY OF THE POSITIVE PREDICTIONS.
RECALL: MEASURES THE ABILITY TO IDENTIFY ALL RELEVANT INSTANCES.
F1-SCORE: HARMONIC MEAN OF PRECISION AND RECALL, PROVIDING A SINGLE METRIC TO EVALUATE PERFORMANCE.
SUPPORT: NUMBER OF TRUE INSTANCES FOR EACH LABEL.

CLASSIFICATION METRICS BY CLASS:

CLASS	PRECISION	RECALL	F1-SCORE	SUPPORT
-1	0.52	0.94	0.67	709
0	0.48	0.16	0.23	606
1	0.69	0.50	0.58	606

MACRO AVERAGE:
PRECISION: 0.56
RECALL: 0.53
F1-SCORE: 0.49

WEIGHTED AVERAGE:
PRECISION: 0.56
RECALL: 0.55
F1-SCORE: 0.50

3. INSIGHTS AND ANALYSIS
HIGH TRAINING ACCURACY VS. LOW TESTING ACCURACY:
THE MODEL PERFORMS EXCEPTIONALLY WELL ON THE TRAINING DATA BUT STRUGGLES ON THE TESTING DATA, INDICATING POTENTIAL OVERFITTING.

CLASS-WISE PERFORMANCE:
CLASS -1 HAS THE HIGHEST RECALL BUT RELATIVELY LOWER PRECISION.
CLASS 0 HAS POOR RECALL AND PRECISION, INDICATING IT IS CHALLENGING FOR THE MODEL TO PREDICT THIS CLASS ACCURATELY.
CLASS 1 HAS A BALANCED PERFORMANCE BUT STILL SHOWS ROOM FOR IMPROVEMENT IN RECALL.

4. CONCLUSION
BEST MODEL PERFORMANCE: AMONG VARIOUS MODELS TESTED, THE RANDOM FOREST MODEL WITH 300 ESTIMATORS AND A MAXIMUM DEPTH OF 20 ACHIEVED THE HIGHEST PERFORMANCE IN TERMS OF TESTING ACCURACY. WHILE IT DEMONSTRATES EXCELLENT TRAINING ACCURACY, THERE IS A NOTABLE GAP IN PERFORMANCE ON UNSEEN DATA, WHICH SUGGESTS OVERFITTING.

CLASS IMBALANCE: THE MODEL SHOWS SIGNIFICANT VARIABILITY IN PERFORMANCE ACROSS DIFFERENT CLASSES, WITH THE LOWEST PERFORMANCE ON CLASS 0. THIS HIGHLIGHTS THE CHALLENGE OF HANDLING CLASS IMBALANCE AND SUGGESTS THE NEED FOR TARGETED IMPROVEMENTS.

THANK YOU