Model Development Phase Template

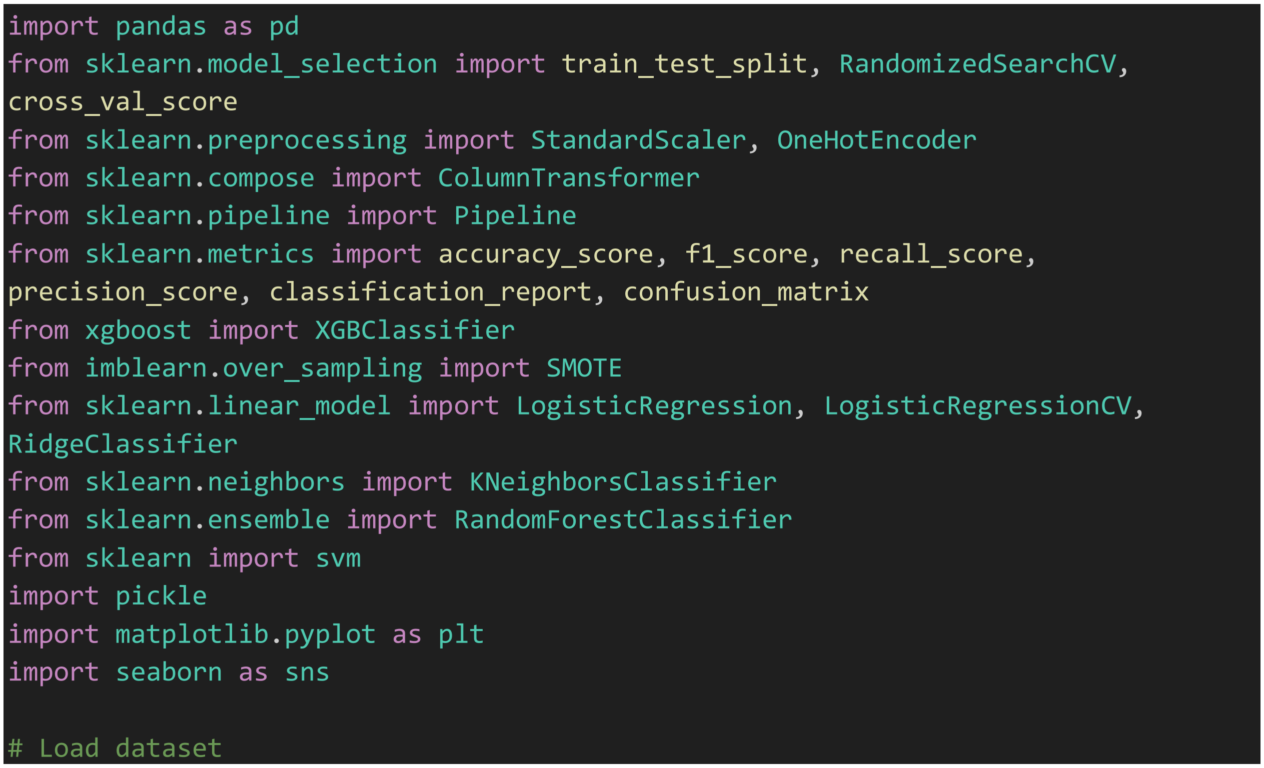
Date	20 July 2024
Team ID	SWTID1720110595
Project Title	Ecommerce Shipping Prediction Using Machine Learning
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

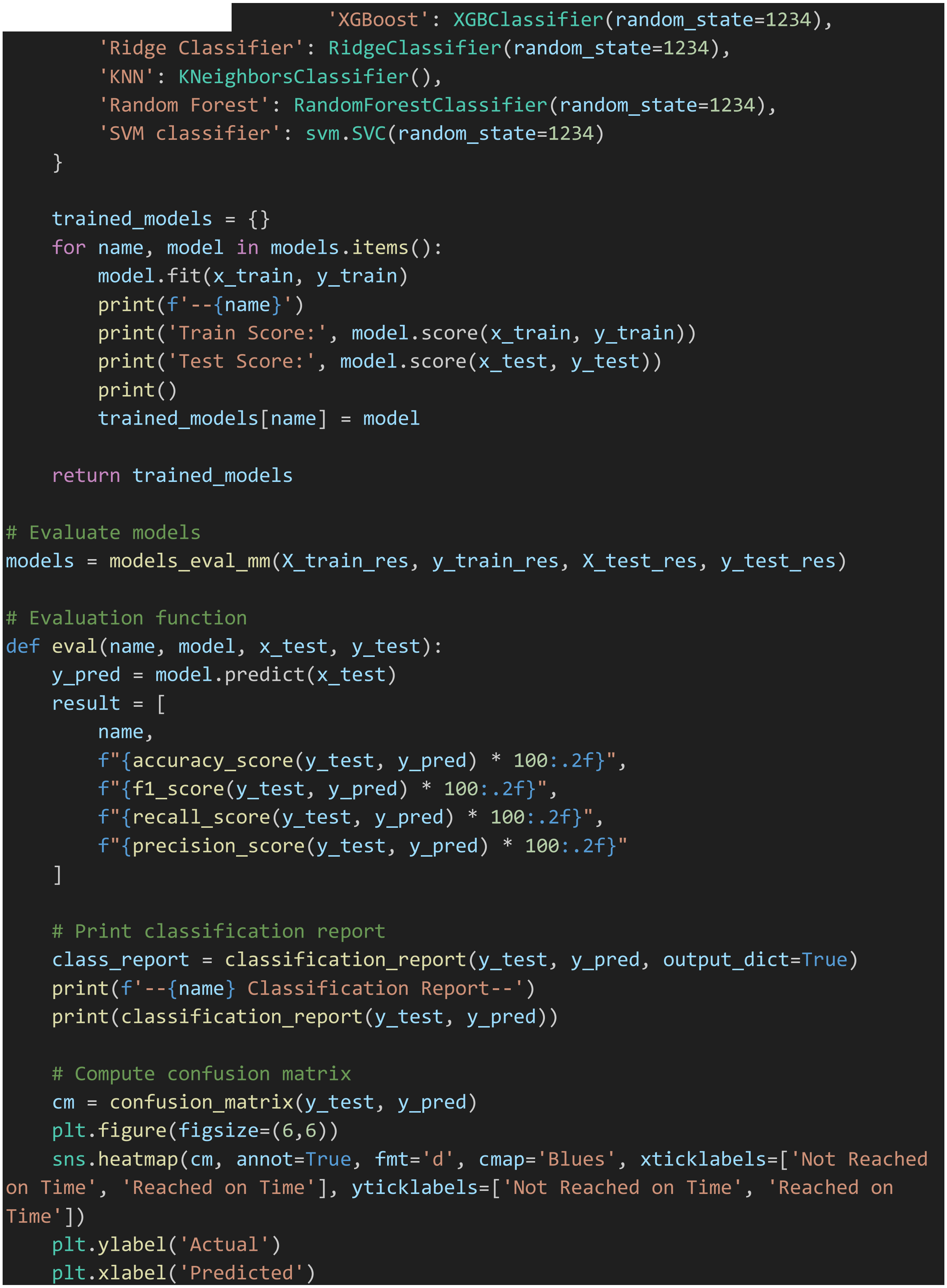
Initial Model Training Code:

Paste the screenshot of the model training code

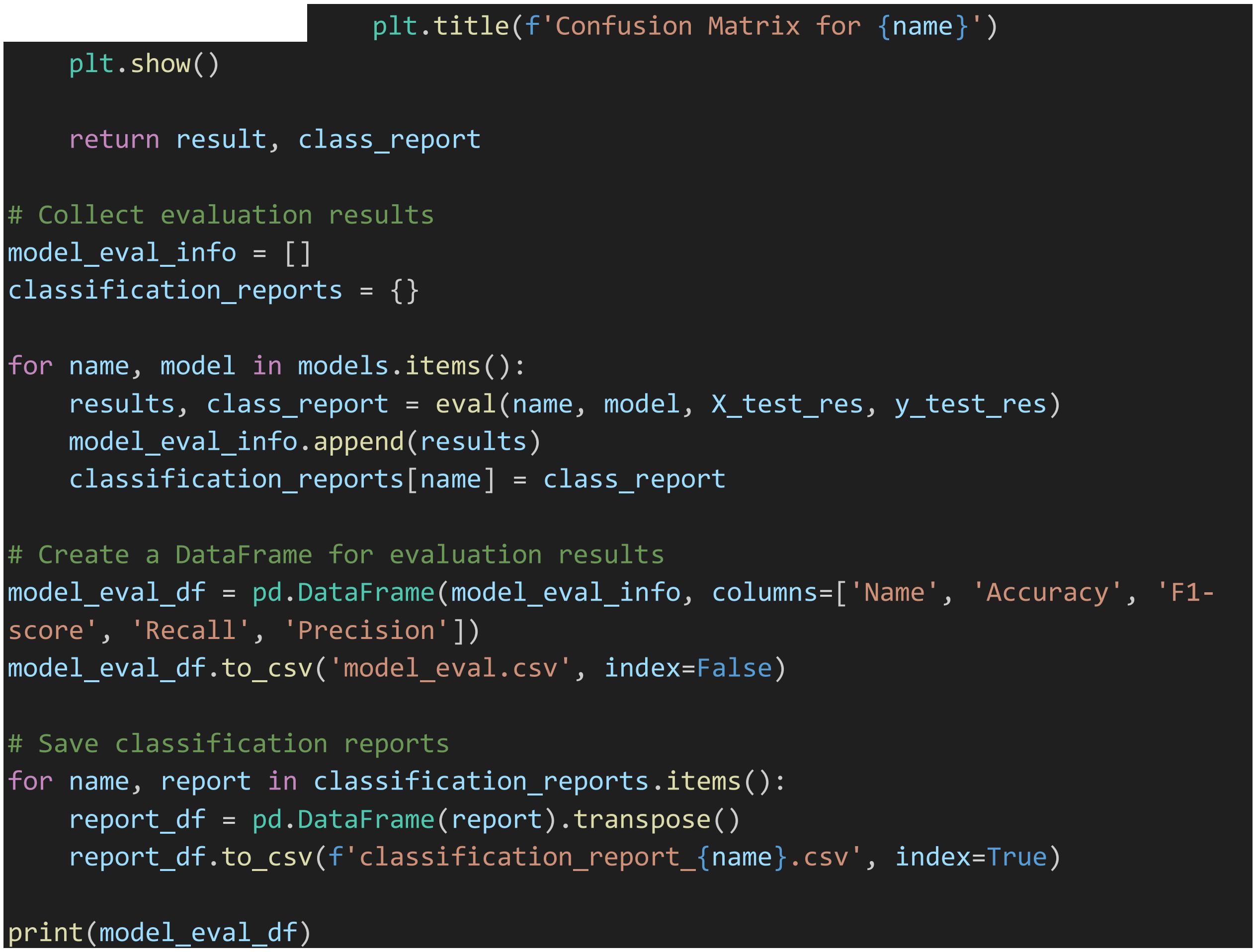


```
import pandas as pd
from sklearn.model_selection import train_test_split, RandomizedSearchCV,
cross_val_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, f1_score, recall_score,
precision_score, classification_report, confusion_matrix
from xgboost import XGBClassifier
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV,
RidgeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
import pickle
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
```

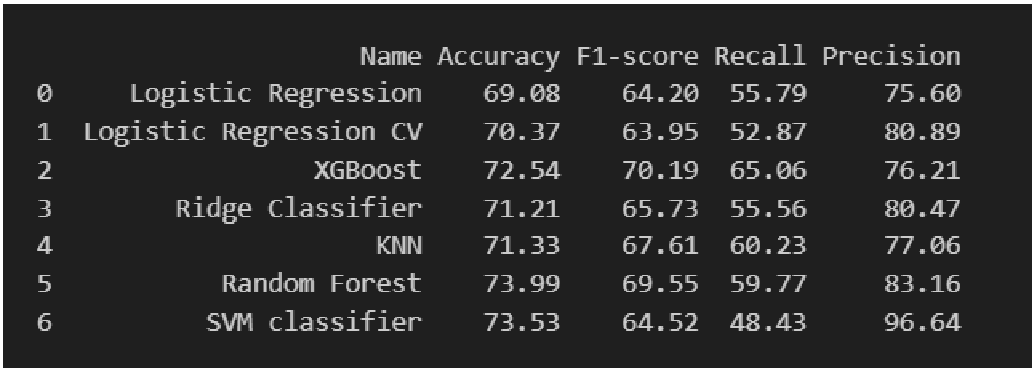
```
data = pd.read_csv('Train.csv')

# Feature engineering
data['Total_Interaction'] = data['Customer_care_calls'] * data['Customer_rating']
data['Cost_per_Weight'] = data['Cost_of_the_Product'] / data['Weight_in_gms']

# Data preprocessing
data = data.fillna(method='ffill')

# Define features and target
X = data.drop(columns=['ID', 'Reached.on.Time_Y.N'])
y = data['Reached.on.Time_Y.N']

# Define numerical and categorical features
numerical_features = ['Customer_care_calls', 'Customer_rating',
'Cost_of_the_Product', 'Prior_purchases', 'Discount_offered', 'Weight_in_gms',
'Total_Interaction', 'Cost_per_Weight']
categorical_features = ['Warehouse_block', 'Mode_of_Shipment',
'Product_importance', 'Gender']

numerical_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore', drop='first')

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# Preprocess data before applying SMOTE
X_preprocessed = preprocessor.fit_transform(X)

# Apply SMOTE to the preprocessed training data
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_preprocessed, y)

# Split resampled data into training and test sets
X_train_res, X_test_res, y_train_res, y_test_res = train_test_split(X_resampled,
y_resampled, test_size=0.2, random_state=42)

# Model evaluation function
def models_eval_mm(x_train, y_train, x_test, y_test):
    models = {
        'Logistic Regression': LogisticRegression(random_state=1234),
        'Logistic Regression CV': LogisticRegressionCV(random_state=1234),
```



```
                            'XGBoost': XGBClassifier(random_state=1234),
        'Ridge Classifier': RidgeClassifier(random_state=1234),
        'KNN': KNeighborsClassifier(),
        'Random Forest': RandomForestClassifier(random_state=1234),
        'SVM classifier': svm.SVC(random_state=1234)
    }

    trained_models = {}
    for name, model in models.items():
        model.fit(x_train, y_train)
        print(f'--{name}')
        print('Train Score:', model.score(x_train, y_train))
        print('Test Score:', model.score(x_test, y_test))
        print()
        trained_models[name] = model

    return trained_models

# Evaluate models
models = models_eval_mm(X_train_res, y_train_res, X_test_res, y_test_res)

# Evaluation function
def eval(name, model, x_test, y_test):
    y_pred = model.predict(x_test)
    result = [
        name,
        f"{accuracy_score(y_test, y_pred) * 100:.2f}",
        f"{f1_score(y_test, y_pred) * 100:.2f}",
        f"{recall_score(y_test, y_pred) * 100:.2f}",
        f"{precision_score(y_test, y_pred) * 100:.2f}"
    ]

    # Print classification report
    class_report = classification_report(y_test, y_pred, output_dict=True)
    print(f'--{name} Classification Report--')
    print(classification_report(y_test, y_pred))

    # Compute confusion matrix
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6,6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Reached
on Time', 'Reached on Time'], yticklabels=['Not Reached on Time', 'Reached on
Time'])
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
```



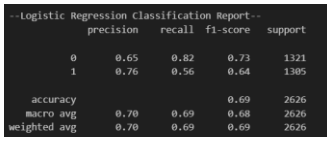
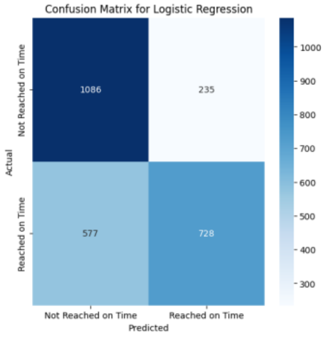
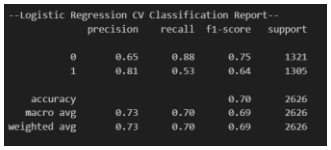
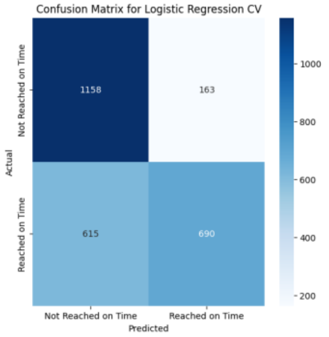
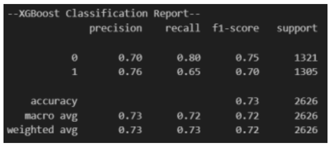
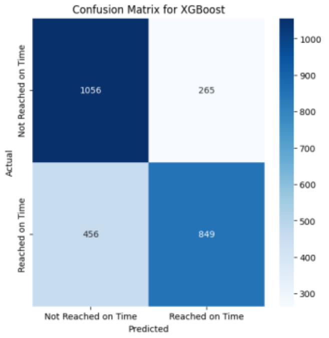
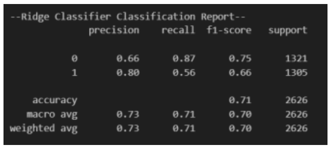
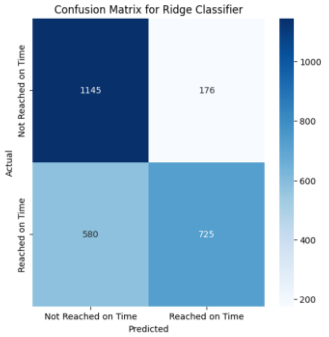
```
    plt.title(f'Confusion Matrix for {name}')
    plt.show()

    return result, class_report

# Collect evaluation results
model_eval_info = []
classification_reports = {}

for name, model in models.items():
    results, class_report = eval(name, model, X_test_res, y_test_res)
    model_eval_info.append(results)
    classification_reports[name] = class_report

# Create a DataFrame for evaluation results
model_eval_df = pd.DataFrame(model_eval_info, columns=['Name', 'Accuracy', 'F1-
score', 'Recall', 'Precision'])
model_eval_df.to_csv('model_eval.csv', index=False)

# Save classification reports
for name, report in classification_reports.items():
    report_df = pd.DataFrame(report).transpose()
    report_df.to_csv(f'classification_report_{name}.csv', index=True)

print(model_eval_df)
```



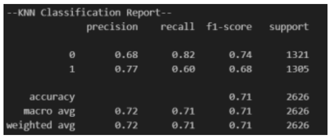
```
                  Name  Accuracy  F1-score  Recall  Precision
0     Logistic Regression     69.08     64.20   55.79      75.60
1  Logistic Regression CV     70.37     63.95   52.87      80.89
2                 XGBoost     72.54     70.19   65.06      76.21
3        Ridge Classifier     71.21     65.73   55.56      80.47
4                     KNN     71.33     67.61   60.23      77.06
5           Random Forest     73.99     69.55   59.77      83.16
6          SVM classifier     73.53     64.52   48.43      96.64
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix

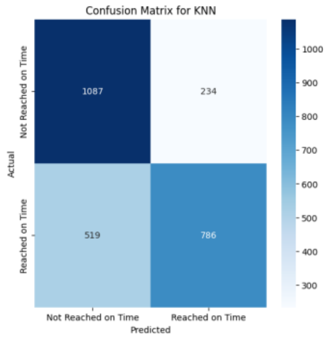
Logistic Regression	 --Logistic Regression Classification Report-- precision recall f1-score support 0 0.65 0.82 0.73 1321 1 0.76 0.56 0.64 1305 accuracy 0.69 2626 macro avg 0.70 0.69 0.68 2626 weighted avg 0.70 0.69 0.69 2626	69	
Model 2	Screenshot of the classification report	Accuracy Value	Screenshot of the confusion matrix
Logistic Regression CV	 --Logistic Regression CV Classification Report-- precision recall f1-score support 0 0.65 0.88 0.75 1321 1 0.81 0.53 0.64 1305 accuracy 0.70 2626 macro avg 0.73 0.70 0.69 2626 weighted avg 0.73 0.70 0.69 2626	70	
XGBoost	 --XGBoost Classification Report-- precision recall f1-score support 0 0.70 0.80 0.75 1321 1 0.76 0.65 0.70 1305 accuracy 0.73 2626 macro avg 0.73 0.72 0.72 2626 weighted avg 0.73 0.73 0.72 2626	73	
Redge Classifier	 --Ridge Classifier Classification Report-- precision recall f1-score support 0 0.66 0.87 0.75 1321 1 0.80 0.56 0.66 1305 accuracy 0.71 2626 macro avg 0.73 0.71 0.70 2626 weighted avg 0.73 0.71 0.70 2626	71	

<table>
<tr>
<td>KNN</td>
<td>

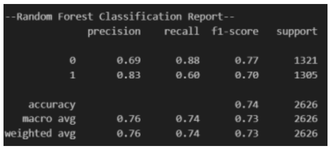


```
--KNN Classification Report--
              precision    recall  f1-score   support

           0       0.68      0.82      0.74      1321
           1       0.77      0.60      0.68      1305

    accuracy                           0.71      2626
   macro avg       0.72      0.71      0.71      2626
weighted avg       0.72      0.71      0.71      2626
```

</td>
<td>71</td>
<td>

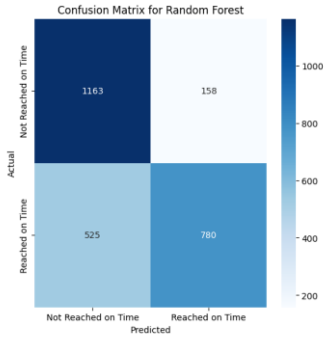


</td>
</tr>
<tr>
<td>Random Forest</td>
<td>

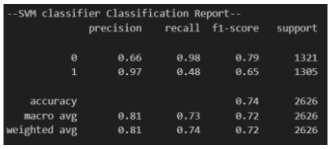


```
--Random Forest Classification Report--
              precision    recall  f1-score   support

           0       0.69      0.88      0.77      1321
           1       0.83      0.60      0.70      1305

    accuracy                           0.74      2626
   macro avg       0.76      0.74      0.73      2626
weighted avg       0.76      0.74      0.73      2626
```

</td>
<td>74</td>
<td>

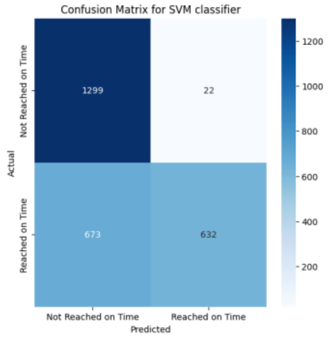


</td>
</tr>
<tr>
<td>SVM Classification</td>
<td>



```
--SVM classifier Classification Report--
              precision    recall  f1-score   support

           0       0.66      0.98      0.79      1321
           1       0.97      0.48      0.65      1305

    accuracy                           0.74      2626
   macro avg       0.81      0.73      0.72      2626
weighted avg       0.81      0.74      0.72      2626
```

</td>
<td>74</td>
<td>



</td>
</tr>
</table>