MACHINE LEARNING - PROJECT

CUSTOMER CHURN PREDICTION "TELECOM SECTOR"

PROBLEM STATEMENT

The current era of telecommunications industry development has transformed global society's behavior in internet usage. The internet is now utilized not only as a communication medium but also as a means of supporting activities, enhancing productivity, and fostering creativity within the community.

This behavior has resulted in a proliferation of telecommunications companies and internet service providers, subsequently intensifying competition among providers.

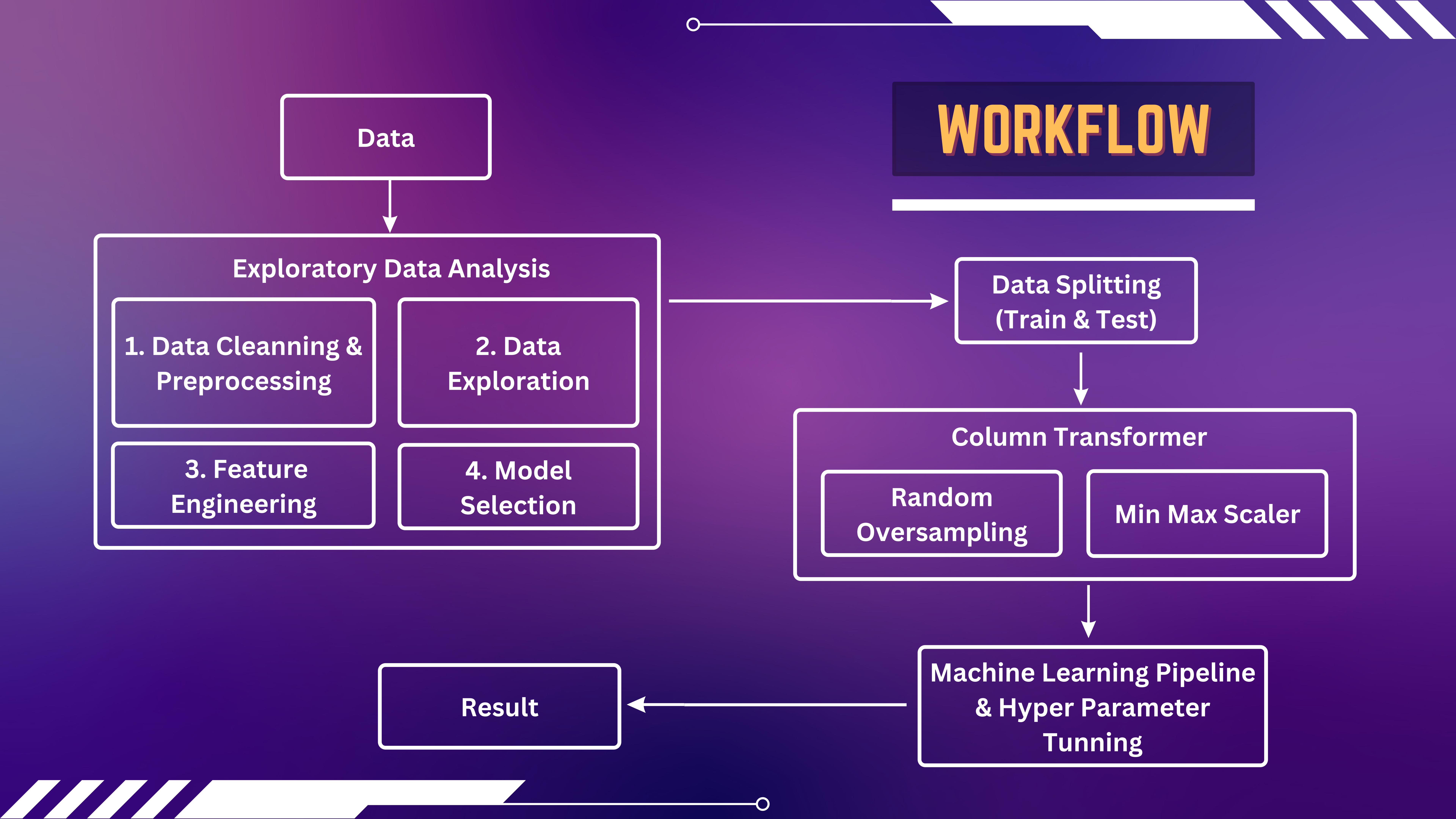
OBJECTTIVE GOAL

Develop a machine learning model capable of classifying customers into two categories: those at risk of churn (unsubscribing) and those who will remain loyal to their subscriptions.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4250 entries, 0 to 4249
Data columns (total 20 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   state                          4250 non-null   object 
 1   account_length                 4250 non-null   int64  
 2   area_code                      4250 non-null   object 
 3   international_plan             4250 non-null   object 
 4   voice_mail_plan                4250 non-null   object 
 5   number_vmail_messages          4250 non-null   int64  
 6   total_day_minutes              4250 non-null   float64
 7   total_day_calls                4250 non-null   int64  
 8   total_day_charge               4250 non-null   float64
 9   total_eve_minutes              4250 non-null   float64
 10  total_eve_calls                4250 non-null   int64  
 11  total_eve_charge               4250 non-null   float64
 12  total_night_minutes            4250 non-null   float64
 13  total_night_calls              4250 non-null   int64  
 14  total_night_charge             4250 non-null   float64
 15  total_intl_minutes             4250 non-null   float64
 16  total_intl_calls               4250 non-null   int64  
 17  total_intl_charge              4250 non-null   float64
 18  number_customer_service_calls  4250 non-null   int64  
 19  churn                          4250 non-null   object 
dtypes: float64(8), int64(7), object(5)
memory usage: 664.2+ KB
```

DATA PREVIEW

The data used is sourced from Kaggle, comprising a total of 4250 rows and 20 columns consisting of numeric and categorical data.


WORKFLOW
Data
Exploratory Data Analysis
1. Data Cleanning & Preprocessing
2. Data Exploration
3. Feature Engineering
4. Model Selection
Data Splitting (Train & Test)
Column Transformer
Random Oversampling
Min Max Scaler
Machine Learning Pipeline & Hyper Parameter Tunning
Result

EXPLORATORY DATA ANALYSIS

1. Data cleanning and preprocessing

In this stage, a recheck is performed to ensure that the available data does not contain null values and that the data type of each column conforms to the appropriate data format.

2. Data exploration

In this stage, data exploration is conducted using various analysis techniques including univariate and bivariate analysis to identify the distribution of the data, the correlation between predictor variables and the target prediction, and other statistical analysis techniques such as finding the weight of evidence, information values, and mutual information. These techniques are useful for identifying the most suitable variables to build a machine learning model.

EXPLORATORY DATA ANALYSIS

3. Feature Engineering

The process of adding, removing, or changing data features aims to more accurately represent the underlying pattern of the data.

4. Model Selection

The selection of a suitable model is crucial in producing the most accurate prediction results.

DATA SPLITTING

The next step in the data processing involves splitting the dataset into two parts: the training data and the testing data. The training data is used for machine learning purposes, while the testing data is used to evaluate the performance of the machine learning model. The dataset is typically split into a 70% portion for training data and a 30% portion for testing data.

COLUMNS TRANSFORMER

In the column transformer stage, the imbalance data is addressed by applying random oversampling techniques to the minority data, ensuring that it has the same proportion as the majority data. Additionally, normalization is performed on the data because certain machine learning algorithm models are sensitive to variations in data ranges. For instance, machine learning models based on gradient descent can be affected by such differences.

MACHINE LEARNING PIPELINE & HYPER PARAMETER TUNNING

1. Machine Learning Pipeline

machine learning pipeline is a structured set of steps or processes to implement and produce an effective machine learning model.

2. Hyper Parameter Tunning

The goal of hyperparameter tuning in machine learning models is to find the optimal combination of hyperparameters that control the behavior and performance of the model.

1. Logistic Regression

```
logreg_pipe = imbPipeline([
    ('scaler', scaler),
    ('oversample', handle_imbalance),
    ('logreg', LogisticRegression())
])

logreg_param = {
    'logreg__penalty': ['l1', 'l2'],
    'logreg__C': [0.1, 1, 10],
    'logreg__solver': ['liblinear', 'saga'],
    'logreg__max_iter': [100, 150, 200],
    'logreg__class_weight': [None, 'balanced']
}

grid_search_logreg = GridSearchCV(estimator=logreg_pipe, param_grid=logreg_param, cv=5, n_jobs=-1, verbose=1)
grid_search_logreg.fit(X_train, y_train)

print('Best parameters :', grid_search_logreg.best_params_)
print('Best Score :', grid_search_logreg.best_score_)

logreg_best_model = grid_search_logreg.best_estimator_
best_model_score = logreg_best_model.score(X_test, y_test)

print('Accuracy : ', best_model_score)
```

The Logistic Regression Model is a simple and easy-to-interpret classification algorithm, suitable for binomial classification with numerical or categorical features.

Despite some drawbacks, such as assumed linearity and limitations for multi-class classification, logistic regression remains a good choice for many classification problems, especially as a baseline for more complex algorithms.

2. Random Forest

```
rf_pipe = imbPipeline([
    ('scaler', scaler),
    ('oversample', handle_imbalance),
    ('rf', RandomForestClassifier())
])

rf_param = {
    'rf__n_estimators': [100, 200, 300],
    'rf__max_depth': [10, 20, 30, None],
    'rf__min_samples_split': [2, 5, 10],
    'rf__min_samples_leaf': [1, 2, 4],
    'rf__bootstrap': [True, False]
}

grid_search_rf = GridSearchCV(estimator=rf_pipe, param_grid=rf_param, cv=5, n_jobs=-1, verbose=1)
grid_search_rf.fit(X_train, y_train)

print("Best param : ", grid_search_rf.best_params_)
print("Best Score : ", grid_search_rf.best_score_)

rf_best_model = grid_search_rf.best_estimator_
best_model_score = rf_best_model.score(X_test, y_test)

print('Accuracy : ', best_model_score)
```

Random Forest is a classification method that combines the strengths of several Decision Trees to produce better predictions. In this process, the algorithm reduces the variance and overfitting that may occur in single Decision Trees.

3. K Nearest Neighbors

```
knn_pipe = imbPipeline([
    ('scaler', scaler),
    ('oversampling', handle_imbalance),
    ('knn', KNeighborsClassifier())]
)

knn_param = {
    'knn__n_neighbors': [5, 7, 10],
    'knn__weights':['uniform', 'distance'],
    'knn__algorithm': ['ball_tree', 'kd_tree', 'brute'],
    'knn__p': [1, 2]
}

grid_search_knn = GridSearchCV(estimator=knn_pipe, param_grid=knn_param, cv=5, verbose=1, n_jobs=-1)
grid_search_knn.fit(X_train, y_train)

print('Best param : ', grid_search_knn.best_params_)
print('Best Score : ', grid_search_knn.best_score_)

knn_best_model = grid_search_knn.best_estimator_

print('Accuracy : ', knn_best_model.score(X_test, y_test))
```

K-Nearest Neighbors (KNN) is a machine learning algorithm used for classification and regression problems. This algorithm works on the "similarity" principle.

In the context of classification, new objects will be classified based on the majority of classes from their k-nearest neighbors. This model is suitable for dealing with non-linear data and a small number of features.

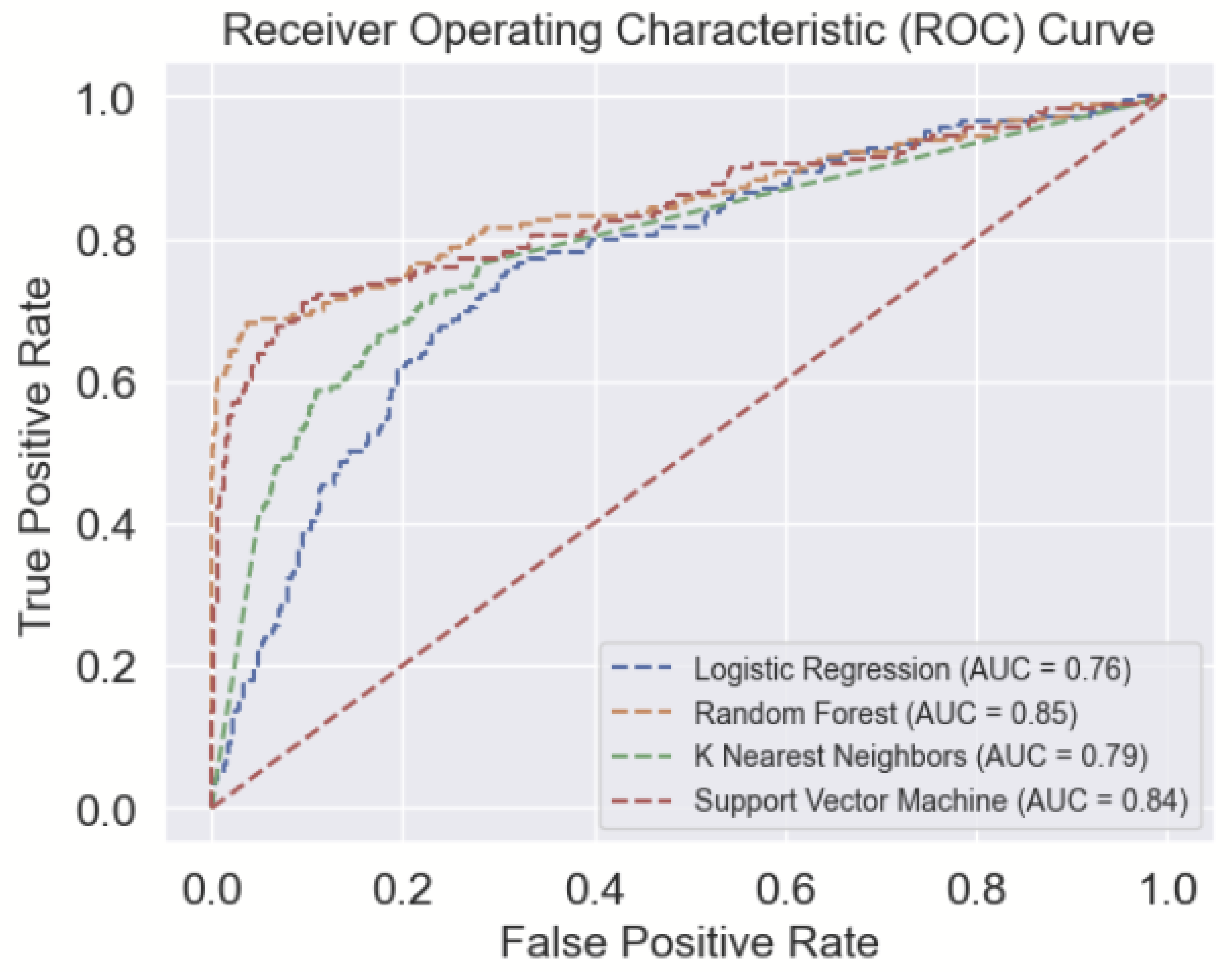
4. Support Vector Machine

```python
svm_pipe = imbPipeline([
    ('scaler', scaler),
    ('oversampling', handle_imbalance),
    ('svm', svm.SVC(probability=True))
])

svm_param = {
    'svm__kernel': ['linear', 'rbf', 'poly'],
    'svm__C': [0.1, 1, 10],
    'svm__gamma': ['scale', 'auto']
}

grid_search_svm = GridSearchCV(estimator=svm_pipe, param_grid=svm_param, cv=5, n_jobs=-1, verbose=1)
grid_search_svm.fit(X_train, y_train)

print("Best param : ", grid_search_svm.best_params_)
print("Best Score : ", grid_search_svm.best_score_)

svm_best_model = grid_search_svm.best_estimator_

print('Accuracy : ', svm_best_model.score(X_test, y_test))
```

Support Vector Machines (SVM) is a powerful and flexible classification algorithm, which works by finding the best hyperplane that separates data into two classes or more. SVM is very effective in solving linear and non-linear classification problems

RESULT

Model	Train Accuracy	Test Accuracy
Logistic Regression	73.07%	71.76%
Random Forest	93.14%	93.56%
K Nearest Neighbours	79.56%	80.31%
Support Vector Machine	87.39%	88.07%



The ROC-AUC graph provides an overall picture of the performance of the classification model. The AUC value reflects the model's ability to distinguish between positive and negative classes. The higher the AUC value, the better the model's performance in distinguishing these classes.

CONCLUSION

Based on the accuracy performance of each machine learning model on the training and testing data, as well as the scores on the ROC AUC chart, it can be concluded that the random forest model exhibits the best performance in predicting customer churn cases. Additionally, with the developed model, predictions can be made on new unlabeled data to determine whether a customer is likely to churn or not churn. This enables management to make business decisions more efficiently and promptly, while also providing appropriate treatment to customers.

THANK YOU

LinkedIn : https://www.linkedin.com/in/adifta-wisnu-wardana/

Project Repositories : https://github.com/nununu-py/FGA-KOMINFO-final-project

Contact : adiftawisnu818@gmail.com