Data Science Part Time 07

Phase Three final project submission

Focus: Classification Analysis

Submitted by: Cynthiah Sheilah Atieno

Instructor name: Samuel Karu

Overview

Business problem

- **Aim:** Syria Telco aims to enhance its customer retention efforts by understanding and predicting customer churn.

- **Goal:** To create a machine learning model that predicts the likelihood of a customer discontinuing their services in the near future.

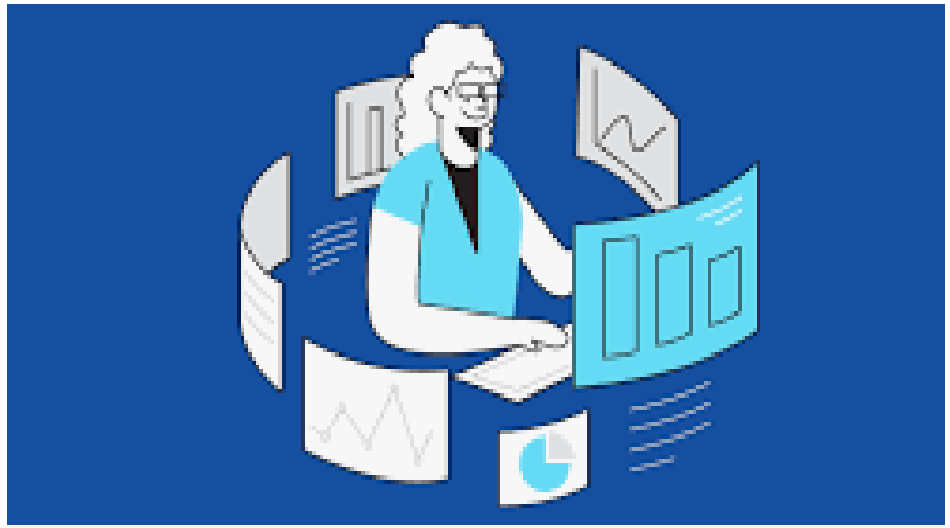
- By identifying high-risk customers early, Syria Telco can tailor personalized retention offers, improve customer satisfaction, and reduce revenue loss associated with churn.

- This model will help the company proactively implement targeted retention strategies, thereby minimizing revenue loss and improving customer retention.

Business understanding: Approach

- Based on the business problem at hand, the steps below were used for data analysis and model creation:
  - Use of Exploratory Data Analysis (EDA) in Python
  - Split the Data into Training and Test Sets
  - Choosing and Training a Model
  - Classification metrics
  - Model Tuning
  - Cross Validation
  - Hyper parameter Optimization

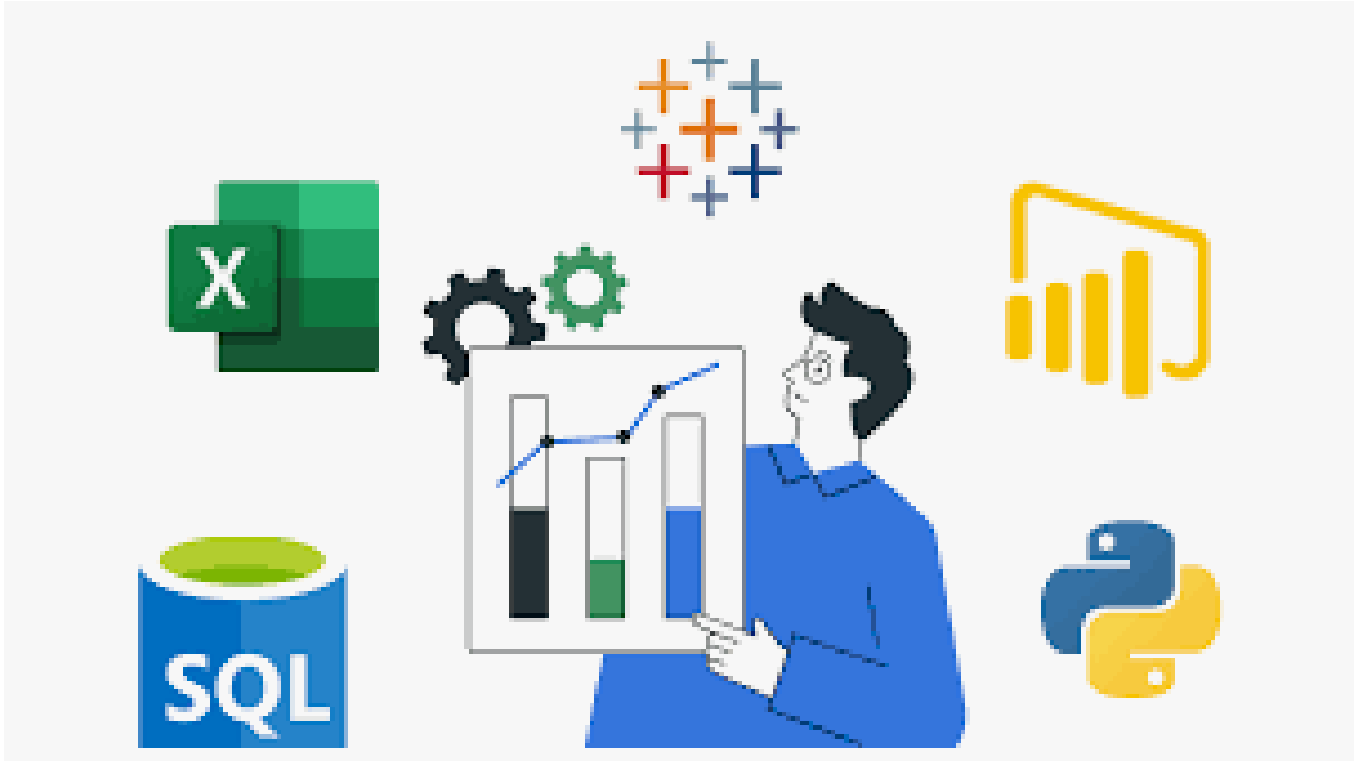
Data understanding



Sets of data:

- Syria Telco data set was provided.
- The primary dataset used for this Analysis is the; 'bigml_59c28831336c6604c800002a.csv' file

Data analysis



Data cleaning and preparation

- Importing relevant libraries
- Loading and previewing the data
- Handling Missing Values
- Checking Number of labels: cardinality
- Encoding Categorical Variables
- Finding the correlation between variables. I used Correlation Heatmap

Step 0: Imports and reading data

- Imported the relevant files to be used in data analysis:
- import numpy as np
- import pandas as pd
- import matplotlib.pyplot as plt
- import seaborn as sns
- from sklearn.preprocessing import StandardScaler
- %matplotlib inline
- from sklearn.model_selection import train_test_split

- from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
- from sklearn.linear_model import LogisticRegression
- from sklearn.metrics import roc_auc_score
- from sklearn.model_selection import cross_val_score, KFold
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.model_selection import StratifiedKFold
- from sklearn.model_selection import GridSearchCV

Step 1: Data understanding

- Viewed different types of rows and columns in the data

- Viewed first five columns in the data frame

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	customer service calls	churn
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78	244.7	91	11.01	10.0	3	2.70	1	False
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62	254.4	103	11.45	13.7	3	3.70	1	False
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6	104	7.32	12.2	5	3.29	0	False
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	196.9	89	8.86	6.6	7	1.78	2	False
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9	121	8.41	10.1	3	2.73	3	False

Step 1: Data understanding

- Described the data frame

	account length	area code	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	customer service calls	
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	200.980348	100.114311	17.083540	200.872037	100.107711	9.039325	10.237294	4.479448	2.764581	1.562856	
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.713844	19.922625	4.310668	50.573847	19.568609	2.275873	2.791840	2.461214	0.753773	1.315491	
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	23.200000	33.000000	1.040000	0.000000	0.000000	0.000000	0.000000	
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.600000	87.000000	14.160000	167.000000	87.000000	7.520000	8.500000	3.000000	2.300000	1.000000	
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.400000	100.000000	17.120000	201.200000	100.000000	9.050000	10.300000	4.000000	2.780000	1.000000	
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.300000	114.000000	20.000000	235.300000	113.000000	10.590000	12.100000	6.000000	3.270000	2.000000	
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.700000	170.000000	30.910000	395.000000	175.000000	17.770000	20.000000	20.000000	5.400000	9.000000	

Step 2: Data preparation

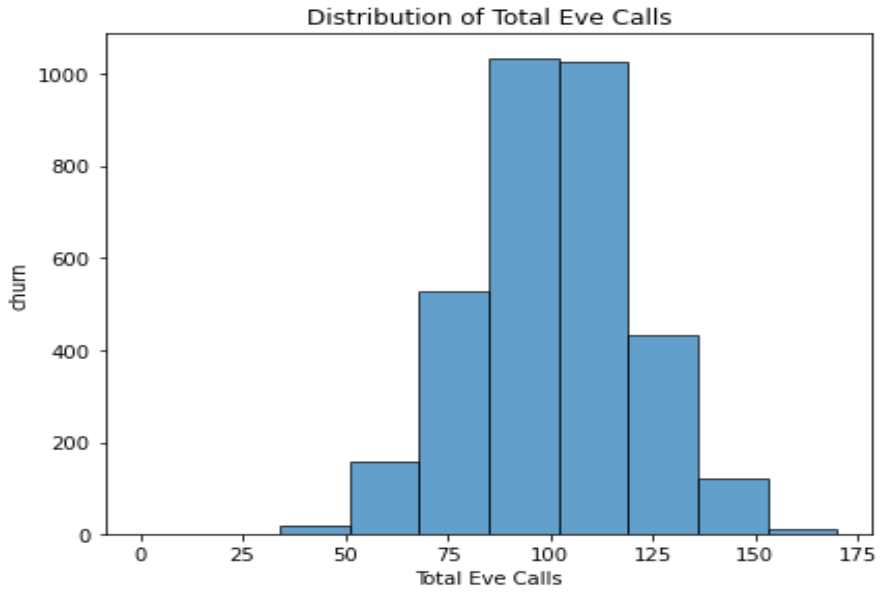
- Key steps followed:
  - Finding missing values
  - Encoding Categorical Variables
  - Feature creation

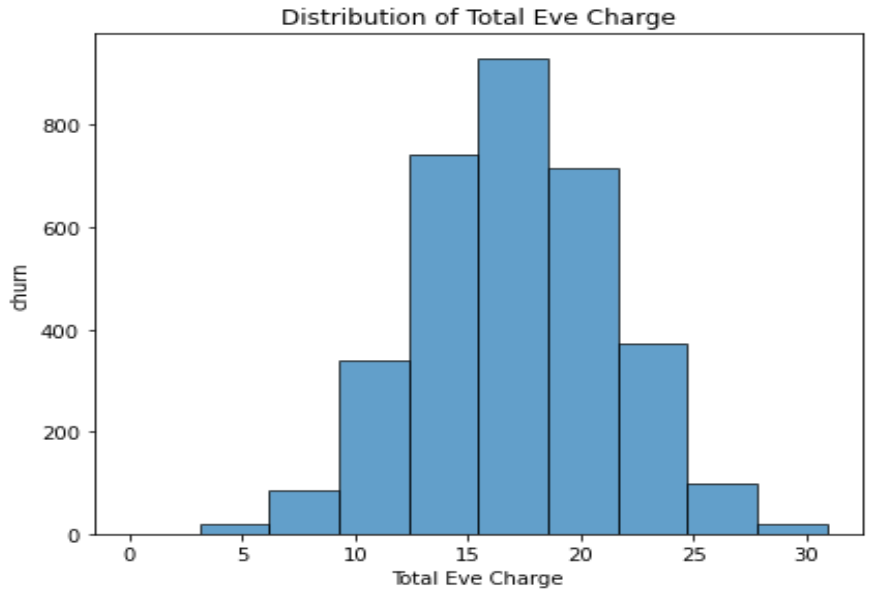
- Summary of categorical variables
  - There are 4 categorical variables. These are given by state, international plan, voice mail plan and phone number.
  - There are two binary categorical variables - international plan and voice mail plan.

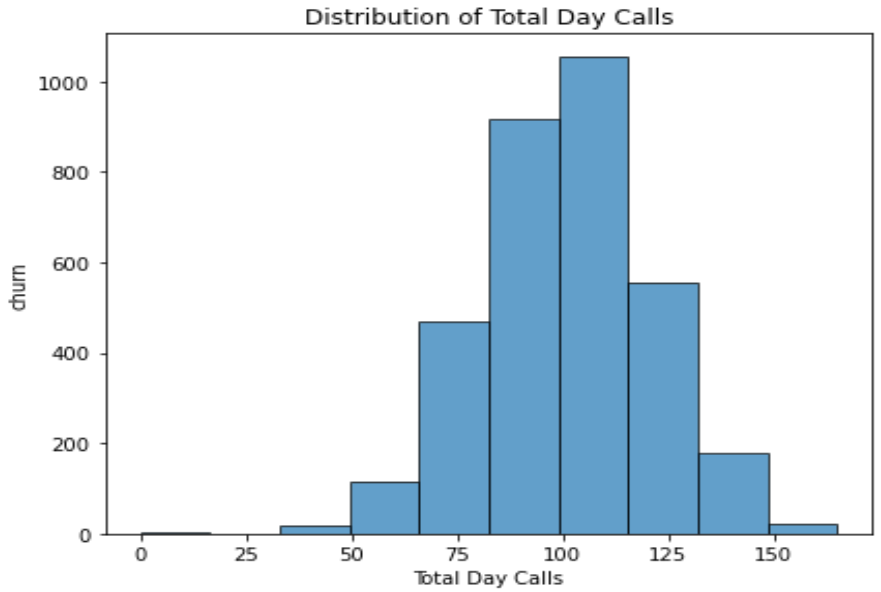
  churn is the target variable.

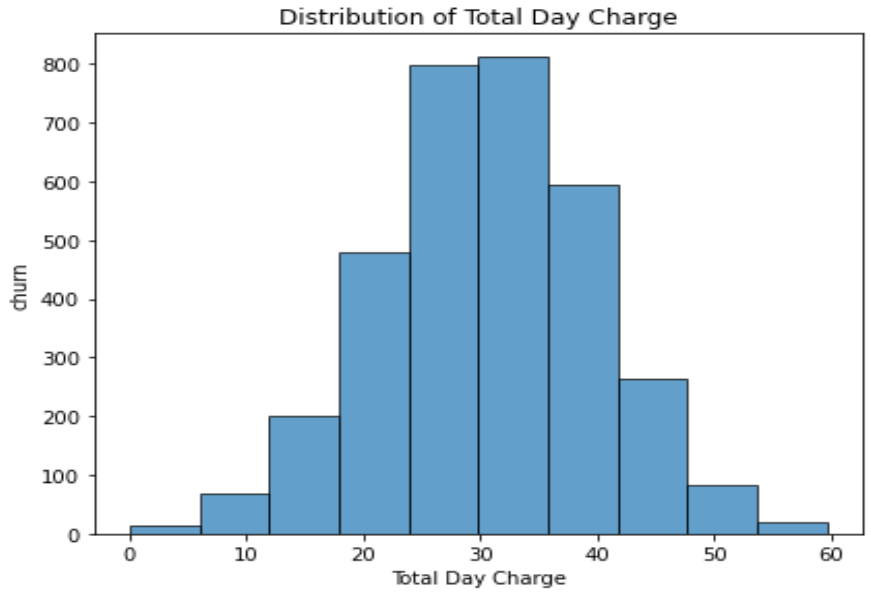
- There were no missing values in categorical variable.

Step 3: Feature understanding





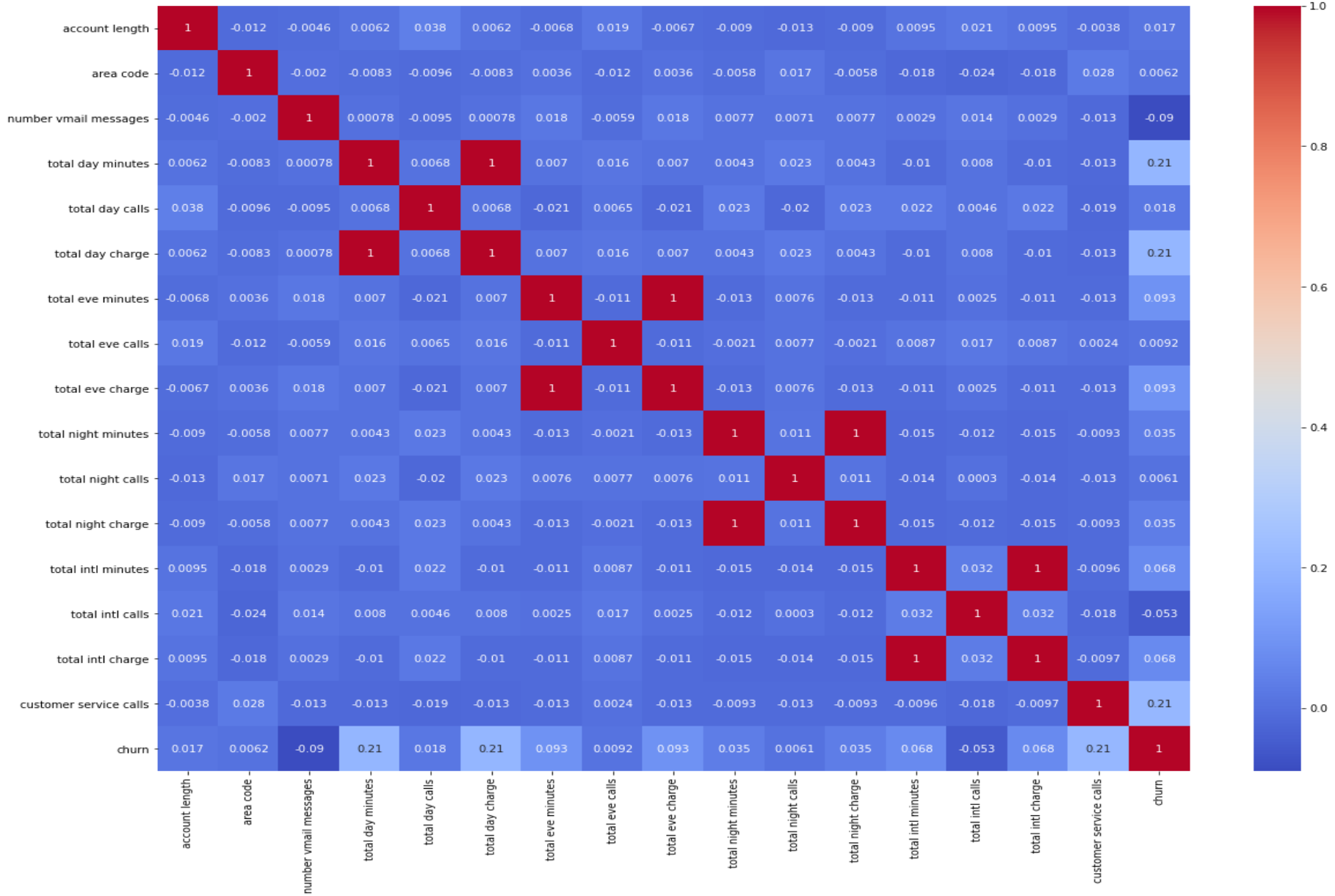




- Key steps followed:
  - Plotting feature distribution
  - Histogram

Step 4: Feature relationships

- Heatmap correlation



Step 5: Exploratory Data Analysis

1. Defining features and target variable

```
X = df_encoded.drop(columns=['churn'])
y = df_encoded['churn']
```

2. Split data into training and test sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

3. Choose and Train a Model

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(len(y_test))
```

Step 5: Exploratory Data Analysis

4. fit the model

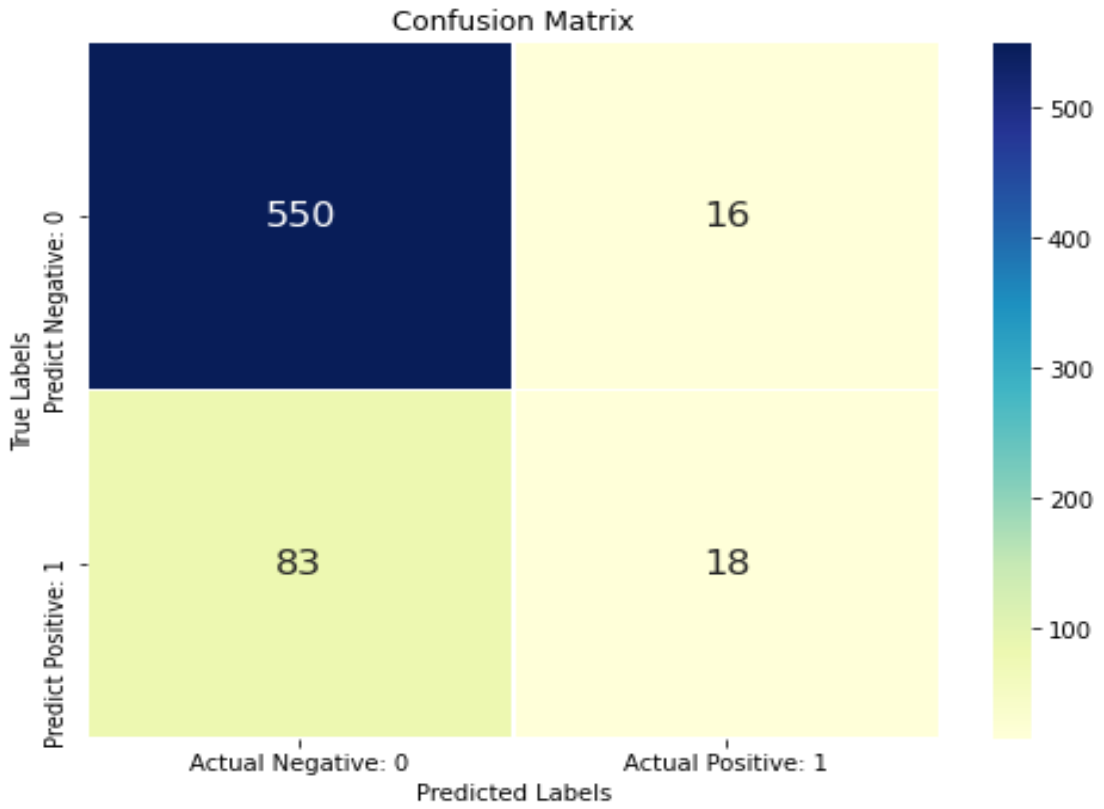
```
model.fit(X_train, y_train)
# Predict on test data
y_pred = model.predict(X_test)
```

5. (a) Compute confusion matrix

```
cm = confusion_matrix(y_test, y_pred)
```

5. (b) Plot the confusion matrix

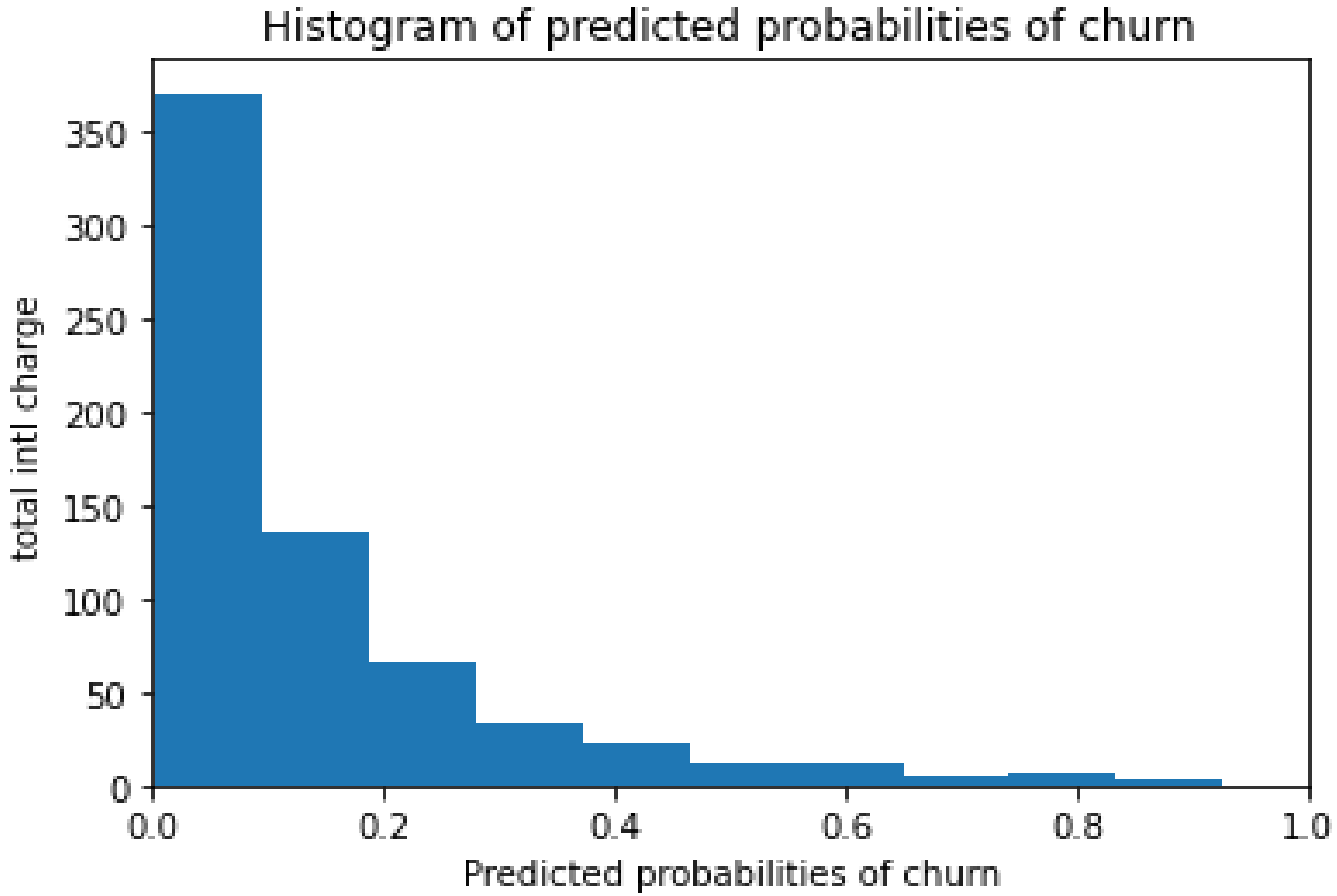
```
plt.figure(figsize=(8, 6))
sns.heatmap(cm_matrix, annot=True, fmt='d',
cmap='YlGnBu',
        cbar=True, annot_kws={"size": 16}, linewidths=.5)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```



Observation

Plot histogram of predicted probabilities

- There are small number of observations with probability > 0.5.
- So, these small number of observations predict that customers will churn.
- Majority of observations predict that customers will not churn.



Model tuning

- I experimented with ROC AUC curve and cross validate with decision tree to find out if they provide a better performance for the True class.

- In this technique, I measured the area under the curve (AUC). A perfect classifier will have a ROC AUC equal to 1, whereas a purely random classifier will have a ROC AUC equal to 0.5.

- So, ROC AUC is the percentage of the ROC plot that is underneath the curve.

Compute ROC AUC

```
ROC_AUC = roc_auc_score(y_test, y_pred1) print('ROC AUC : {:.4f}'.format(ROC_AUC))
```

- ROC AUC of our model approaches towards 1. So, we can conclude that our classifier does a good job in predicting whether customer will churn or not.

Cross validation

- I used decision tree for cross validation
- Initialized the Decision Tree classifier

```
dt_classifier = DecisionTreeClassifier()
```

Define Cross-Validation Strategy

- I used StratifiedKFold as it ensures each fold has the same proportion of class labels, which is especially useful for classification problems.
- Define cross-validation strategy

```
cv = StratifiedKFold(n_splits=5, shuffle=True,
random_state=42)
```

Apply cross-validation

```
scores = cross_val_score(dt_classifier, X, y, cv=cv,
scoring='accuracy')
```

Print results

```
print(f"Accuracy scores for each fold: {scores}")
print(f"Mean accuracy: {scores.mean():.2f}")
print(f"Standard deviation of accuracy: {scores.std():.2f}")
```

Cross validation: Findings

- Our, original model score is found to be 0.852. The average cross-validation score is 0.91.
- The mean accuracy of 0.91 indicates that, on average, the decision tree model correctly classifies 91% of the samples across all folds. This is a high accuracy, suggesting that the model performs well on the dataset.
- Standard deviation: 0.02
- The standard deviation of 0.02 indicates that the accuracy scores vary slightly across the folds. This low standard deviation suggests that the model's performance is consistent and stable across different subsets of the data

Hyperparameter Optimization using GridSearch CV

```
Define parameter grid
param_grid = {
   'criterion': ['gini', 'entropy'],
   'max_depth': [None, 10, 20, 30],
   'min_samples_split': [2, 5, 10],
   'min_samples_leaf': [1, 2, 4]
```

- Initialize GridSearchCV

```
grid_search = GridSearchCV(
   estimator=dt_classifier,
   param_grid=param_grid,
   cv=5,  # Number of cross-validation folds
   scoring='accuracy',  # Metric to evaluate performance
   n_jobs=-1,  # Number of parallel jobs to run (-1 means using all processors)
   verbose=1  # Verbosity level
```

Hyperparameter Optimization using GridSearch CV

- Fit the model with GridSearchCV

```
grid_search.fit(X_train, y_train)
```

- Get the best parameters and best score

```
print(f"Best parameters: {grid_search.best_params_}")
print(f"Best score: {grid_search.best_score_:.2f}")
```

- Get the best estimator

```
best_model = grid_search.best_estimator_
```

- Predict using the best model

```
y_pred = best_model.predict(X_test)
```

- Evaluate the model

```
print(f"Test accuracy: {accuracy_score(y_test,
y_pred):.2f}")
```

Findings:

Our original model test accuracy is 0.852 while GridSearch CV accuracy is 0.94.

We can see that GridSearch CV improve the performance for this particular model.

Overall Findings

- From the analysis of the Syria Telco data sets, below are the ***findings***:

1. The logistic regression model accuracy score is 0.852. So, the model does a very good job in predicting whether or not the customer will churn.

2. Small number of observations predict that customer will churn. Majority of observations predict that customer will not churn.

3. Our original model test accuracy is 0.852 while GridSearch CV accuracy is 0.94. We can see that GridSearch CV improve the performance for this particular model.

Recommendations

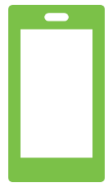
- Leverage the Best-Performing Model
  - Adopt the Decision Tree model for primary churn predictions due to its higher accuracy and consistency.
- Address Class Imbalance: There is a small number of observations predicting churn, indicating class imbalance. I would recommend the use of resampling techniques or class weight adjustments
- Validate Model Performance
  - Monitor and validate models regularly to ensure continued performance and adaptation.

- Leverage GridSearch CV to optimize both models and refine feature selection.
- Business Implications and Actions Implement actionable strategies based on model predictions to reduce churn effectively. Churn Prediction Utilization: Use the high-performing models to proactively address customer churn. Implement strategies such as targeted retention campaigns, personalized offers, or enhanced customer service for those predicted to churn.
- Resource Allocation: Allocate resources effectively based on model predictions to maximize the impact on customer retention.

Any questions?

Thank you so much

Cynthiah Atieno



+254 700 065572

cynthia.atieno@student.moringaschool.com