PROJECT REPORT

TELECOM CHURN ANALYSIS

Lambton College

SUBMITTED TO:
PROFESSOR MEYSAM EFFATI

SUBMITTED BY:
GROUP H
ROHIT KUMAR(C0895100)
TANVI TAKKAR(C0907019)
REWANT SHARMA(C0894265)
NIPUN GULATI(C0896042)
SANIKA SABEESH(C0900949)
ISHIKA SUKHIJA(C0895302)

Table of Contents

- Analysis of Strengths and Weaknesses
- Explanation of Unexpected Outcomes or Observations
- Comparison with Prior Work and Contribution to Existing Knowledge

7. Conclusion
    - Summary of Key Findings
    - Achievement of Project Objectives
    - Recommendations for Future Work or Areas for Improvement
8. References

1.Abstract:

Objectives:

This project's main goal was to create a predictive model that might identify telecom consumers who might be at risk of leaving the company. The project's goal was to lower customer attrition and enable proactive retention methods by attaining this goal.

Methods Used:

The research made use of a number of machine learning algorithms, such as gradient boosting machines, decision trees, random forests, logistic regression, and an ensemble voting method. These algorithms were used for feature engineering, model training, evaluation, and preprocessing of data.

Key Findings:

By means of thorough data analysis and model creation, the study revealed essential information about churn trends in the telecom customer market. Important discoveries included determining important churn predictors, comprehending the connections between distinct characteristics and churn results, and assessing how well various machine learning methods performed.

2.Introduction:

Background:

A major obstacle to customer retention for telecommunications firms is the intense competition and changing preferences of consumers. For these businesses, customer attrition, or churn, is a major worry.

Problem Statement:

To support proactive retention tactics, the project seeks to address the requirement for accurate churn prediction.

Objectives:

The main goal is to create a predictive model to identify consumers who are at danger of leaving. Data collection, preprocessing, model creation, and evaluation are all part of the methodology.

Methodology:

Algorithms Used:

Various machine learning algorithms such as [list algorithms] were employed for churn prediction.

Justification:

The choice of algorithms was based on their suitability for the problem domain, scalability, and interpretability.

Training and Evaluation:

Models were trained on the preprocessed data and evaluated using metrics like accuracy, precision, recall, and AUC-ROC.

Parameter Tuning:

Hyperparameter tuning techniques were applied to optimise model performance.

3. Data Collection and Preprocessing

Data Reading

Reading and saving the data in a Pandas data frame using the pandas.read_csv method.

```
df=pd.read_excel('/content/Telecom Churn Rate Dataset (1).xlsx')
```

```
[95] df.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	Month-to-month	Yes	Electronic check	29.85	29.85
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	No	No	One year	No	Mailed check	56.95	1889.5
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	Month-to-month	Yes	Mailed check	53.85	108.15
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	Month-to-month	Yes	Electronic check	70.70	151.65

5 rows × 23 columns

Basic Information of the dataset

- customerID: Customer ID
- genderCustomer: gender (female, male)
- SeniorCitizen: Whether the customer is a senior citizen or not (1, 0)
- PartnerWhether: the customer has a partner or not (Yes, No)
- Dependents: Whether the customer has dependents or not (Yes, No)
- tenure: Number of months the customer has stayed with the company
- PhoneService: Whether the customer has a phone service or not (Yes, No)
- MultipleLines: Whether the customer has multiple lines or not (Yes, No, No phone service)
- InternetService: Customer’s internet service provider (DSL, Fiber optic, No)
- OnlineSecurity: Whether the customer has online security or not (Yes, No, No internet service)
- OnlineBackup: Whether the customer has online backup or not (Yes, No, No internet service)
- DeviceProtection: Whether the customer has device protection or not (Yes, No, No internet service)

- TechSupport: Whether the customer has tech support or not (Yes, No, No internet service)
- StreamingTV: Whether the customer has streaming TV or not (Yes, No, No internet service)
- StreamingMovies: Whether the customer has streaming movies or not (Yes, No, No internet service)
- Contract: The contract term of the customer (Month-to-month, One year, Two year)
- PaperlessBilling: Whether the customer has paperless billing or not (Yes, No)
- PaymentMethod: The customer's payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))
- MonthlyCharges: The amount charged to the customer monthly
- TotalCharges: The total amount charged to the customer
- numAdminTickets: The number of Admin Ticket
- numTechTickets: The number of Tech Ticket
- Churn: Whether the customer churned or not (Yes or No)

2. Exploratory Data Analysis and Data Cleaning

Exploratory data analysis is the process of examining a data set's primary features, typically with the use of summary statistics and visualization techniques. The aim is to comprehend the data, identify trends and irregularities, and verify assumptions prior to carrying out more assessments.

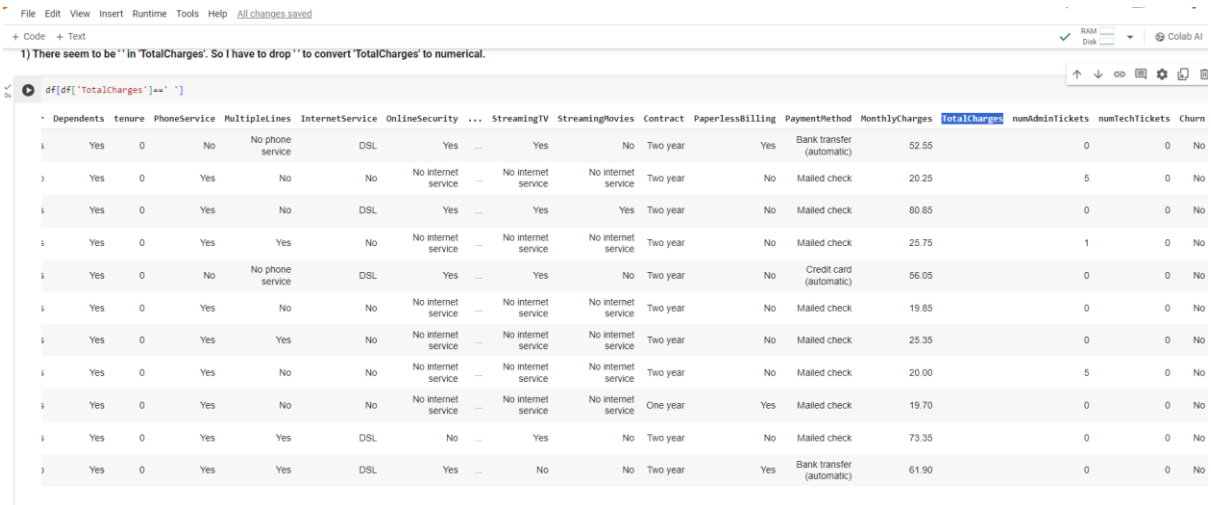
Missing Values and Data Types

Pandas are used in the early stages of EDA when we want to learn as much as we can about the data. It is useful to use the DataFrame.info function. The number of non-null values, the column names and their data types, the amount of memory utilised by the data frame, and other pertinent information are printed in a clear summary of the data frame by using this technique.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 23 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  numAdminTickets   7043 non-null   int64  
 21  numTechTickets    7043 non-null   int64  
 22  Churn             7043 non-null   object 
dtypes: float64(1), int64(4), object(18)
memory usage: 1.2+ MB
```

The dataset consists of 7,043 entries with 22 different features. Notably, there are no missing values in the dataset. However, upon examination, it's evident that the 'TotalCharges' column, intended to represent the total amount charged to customers, has been incorrectly categorized as an object datatype. To facilitate further analysis, it's imperative to convert this column to a numeric datatype. Consequently, prior to conversion, these empty strings will be removed from the dataset. This step ensures the integrity of the data transformation process and facilitates accurate numerical analysis.


File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

1) There seem to be ' ' in 'TotalCharges'. So I have to drop ' ' to convert 'TotalCharges' to numerical.

```
df[df['TotalCharges']==' ']
```

	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	numAdminTickets	numTechTickets	Churn
	Yes	0	No	No phone service	DSL	Yes	...	Yes	No	Two year	Yes	Bank transfer (automatic)	52.55		0	0	No
	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	Two year	No	Mailed check	20.25		5	0	No
	Yes	0	Yes	No	DSL	Yes	...	Yes	Yes	Two year	No	Mailed check	80.85		0	0	No
	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No internet service	Two year	No	Mailed check	25.75		1	0	No
	Yes	0	No	No phone service	DSL	Yes	...	Yes	No	Two year	No	Credit card (automatic)	56.05		0	0	No
	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	Two year	No	Mailed check	19.85		0	0	No
	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No internet service	Two year	No	Mailed check	25.35		0	0	No
	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	Two year	No	Mailed check	20.00		5	0	No
	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	One year	Yes	Mailed check	19.70		0	0	No
	Yes	0	Yes	Yes	DSL	No	...	Yes	No	Two year	No	Mailed check	73.35		0	0	No
	Yes	0	Yes	Yes	DSL	Yes	...	No	No	Two year	Yes	Bank transfer (automatic)	61.90		0	0	No

Checking whether there are duplicates or not

2) If there are some duplicated liness, I have to drop them.

```
[100] df.duplicated().sum()
```

```
0
```

Remove off the customerID column.

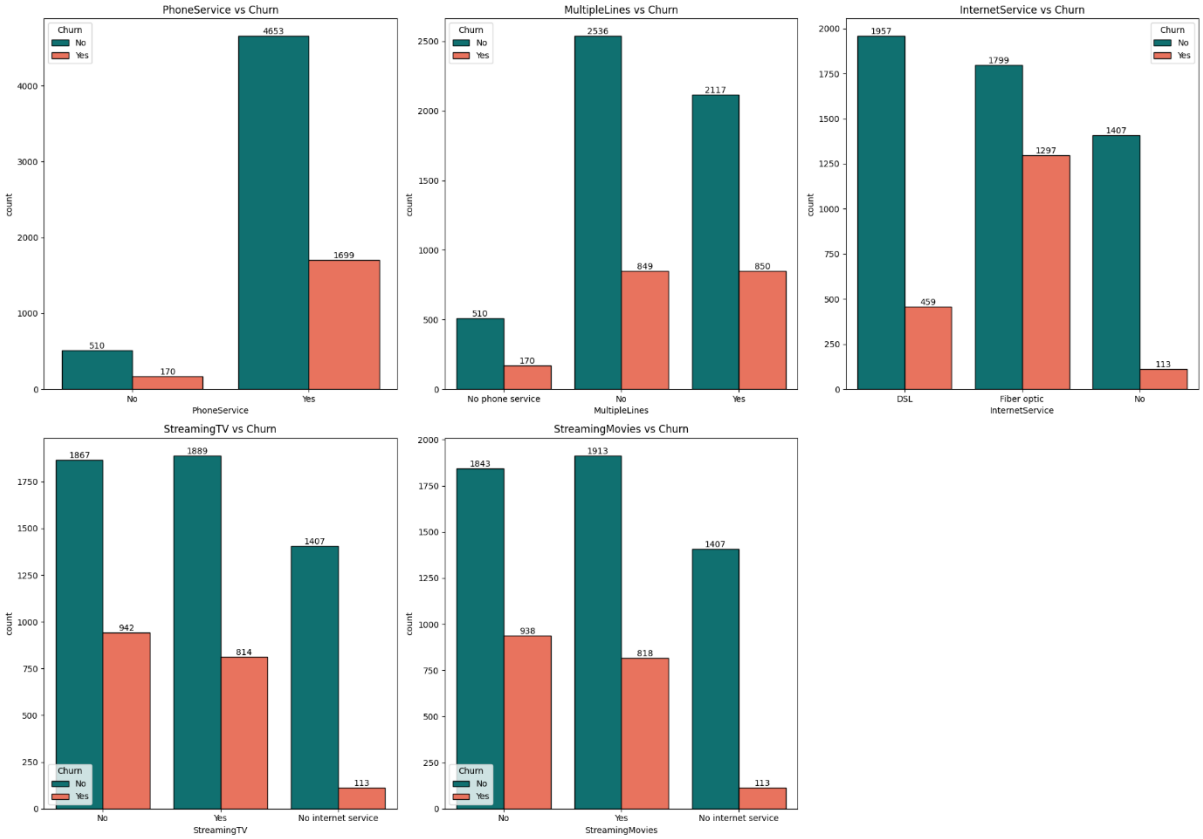
To determine whether a client would churn, the customerID field is meaningless. We remove this column from the data set as a result.

3) Dropping 'customerID'

```
df = df.drop('customerID', axis=1)
```

Visualizations

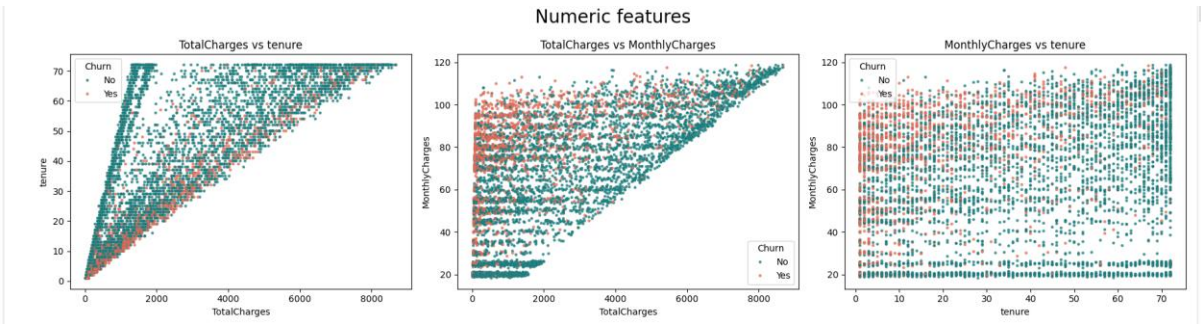
Provided services by Churn and not Churn



Observations:

Customers predominantly stayed with phone service, with fewer leaving, especially among those without it. Multiple lines users had varied retention, while non-users experienced higher churn. Fiber optic internet users churned the most, while DSL users had lower churn. Non-users of streaming services exhibited lower churn compared to users.

Scatter plot by Total Charge, Monthly Charge and Tenure



Observations:

- **Total Charges vs. Tenure**: Higher tenure seems to correlate with higher total charges, indicating long-term customers tend to spend more over time.

- **Monthly Charges vs. Tenure**: There is a wide distribution of monthly charges across different tenures, suggesting a variety of plans and customer preferences.
- **Churn Indication**: Data points are color-coded to show churn, with a noticeable cluster of churned customers at lower tenure and higher monthly charges, implying that new customers with higher monthly costs are more likely to churn.

Feature Engineering

The process of removing features from the data and formatting them such that the machine learning model can use them is known as feature engineering. Both numerical and categorical variables need to be transformed for this project. All of the dataset's categorical attributes should be converted into numerical labels before the model is trained since the majority of machine learning methods need numerical values. We also need to convert the numerical columns into a similar scale. By doing this, the learning process will not be dominated by columns with high values. Below is a more detailed description of the approaches used in this project. We use solely Pandas to do all transformations, but we also offer an alternate approach that makes use of Scikit-Learn.

No modification required in the SeniorCitizen column. It is already a binary column and should not be modified.

Label Encoding

Label encoding is a technique utilized to substitute categorical values with corresponding numerical labels. This method assigns a unique numerical identifier to each category, effectively transforming categorical data into a numerical format.

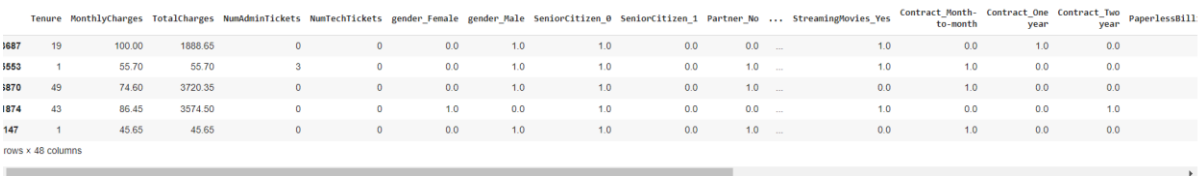
5) Label Encoding

```
# Label Encoding refers to converting the labels into a numeric form.
# This is only for EDA reasons. Later we will use OneHotEncoder to prepare for model building.
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df1 = df.copy()

df1[categoric_columns] = df1[categoric_columns].apply(le.fit_transform)
df1[['Churn']] = df1[['Churn']].apply(le.fit_transform)
```

One hot Encoder

Every level of the category variable has a new binary column created using one-hot encoding. The zeros and ones in the new column denote whether the category is present in the data or not. We use one-hot encoding for the following categorical variables in our project: 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',

'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies','Contract', 'PaperlessBilling', 'PaymentMethod'



	Tenure	MonthlyCharges	TotalCharges	NumAdminTickets	NumTechTickets	gender_Female	gender_Male	SeniorCitizen_0	SeniorCitizen_1	Partner_No	...	StreamingMovies_Yes	Contract_Month-to-month	Contract_One year	Contract_Two year	PaperlessBill
687	19	100.00	1888.65	0	0	0.0	1.0	1.0	0.0	0.0	...	1.0	0.0	1.0	0.0	
553	1	55.70	55.70	3	0	0.0	1.0	1.0	0.0	1.0	...	1.0	1.0	0.0	0.0	
870	49	74.60	3720.35	0	0	0.0	1.0	1.0	0.0	1.0	...	0.0	1.0	0.0	0.0	
874	43	86.45	3574.50	0	0	1.0	0.0	1.0	0.0	0.0	...	1.0	0.0	0.0	1.0	
147	1	45.65	45.65	0	0	0.0	1.0	1.0	0.0	1.0	...	0.0	1.0	0.0	0.0	

rows × 48 columns

5.Methodology

Feature Importance

Feature importance is determined through mutual information, which assesses the level of interdependence between two variables using entropy estimations. In machine learning, our focus lies in assessing the extent of dependency between each independent variable and the response variable. Elevated mutual information values indicate a stronger dependency, suggesting that the independent variable holds significance in predicting the target variable.

1) Features importance

```
print(X_train.columns)
```

```
Index(['tenure', 'MonthlyCharges', 'TotalCharges', 'numAdminTickets',
       'numTechTickets', 'onehotencoder__gender_Female',
       'onehotencoder__gender_Male', 'onehotencoder__SeniorCitizen_0',
       'onehotencoder__SeniorCitizen_1', 'onehotencoder__Partner_No',
       'onehotencoder__Partner_Yes', 'onehotencoder__Dependents_No',
       'onehotencoder__Dependents_Yes', 'onehotencoder__PhoneService_No',
       'onehotencoder__PhoneService_Yes', 'onehotencoder__MultipleLines_No',
       'onehotencoder__MultipleLines_No phone service',
       'onehotencoder__MultipleLines_Yes',
       'onehotencoder__InternetService_DSL',
       'onehotencoder__InternetService_Fiber optic',
       'onehotencoder__InternetService_No', 'onehotencoder__OnlineSecurity_No',
       'onehotencoder__OnlineSecurity_No internet service',
       'onehotencoder__OnlineSecurity_Yes', 'onehotencoder__OnlineBackup_No',
       'onehotencoder__OnlineBackup_No internet service',
       'onehotencoder__OnlineBackup_Yes', 'onehotencoder__DeviceProtection_No',
       'onehotencoder__DeviceProtection_No internet service',
       'onehotencoder__DeviceProtection_Yes', 'onehotencoder__TechSupport_No',
       'onehotencoder__TechSupport_No internet service',
       'onehotencoder__TechSupport_Yes', 'onehotencoder__StreamingTV_No',
       'onehotencoder__StreamingTV_No internet service',
       'onehotencoder__StreamingTV_Yes', 'onehotencoder__StreamingMovies_No',
       'onehotencoder__StreamingMovies_No internet service',
       'onehotencoder__StreamingMovies_Yes',
       'onehotencoder__Contract_Month-to-month',
       'onehotencoder__Contract_One year', 'onehotencoder__Contract_Two year',
       'onehotencoder__PaperlessBilling_No',
       'onehotencoder__PaperlessBilling_Yes',
       'onehotencoder__PaymentMethod_Bank transfer (automatic)',
       'onehotencoder__PaymentMethod_Credit card (automatic)',
       'onehotencoder__PaymentMethod_Electronic check',
       'onehotencoder__PaymentMethod_Mailed check'],
      dtype='object')
```

Splitting the data in training and testing sets

When building a model, the initial stage involves dividing the data into two distinct sets: the training set and the testing set. The training set serves as the foundation for the machine learning algorithm to construct the model. On the other hand, the test set comprises samples independent of the learning process and is utilized to gauge the model's performance. Evaluating the model's quality with unseen data ensures an unbiased assessment, validating its effectiveness beyond the training data.
Initially, we define a variable X to hold the independent attributes of the dataset. Furthermore, a variable y is created specifically to store the target variable (Churn).

Algorithm Selected:

1. Random Forest and RandomForestClassifier: These are ensemble learning methods that construct a multitude of decision trees during training and output the mode of the classes for classification tasks (RandomForestClassifier specifically). They are robust and effective for handling complex datasets.

2. K-Fold Cross-validation: This technique partitions the dataset into K subsets and performs training and validation K times, using each subset as the validation set once. This helps in obtaining more reliable model performance estimates, especially when the dataset is limited.

3. XGBoost: XGBoost is a gradient boosting algorithm known for its efficiency and high performance. It builds trees sequentially, optimizing the loss function at each step, making it suitable for classification tasks like predicting telecom churn.

4. Hyperparameter Tuning Using GridSearchCV: GridSearchCV exhaustively searches through a specified parameter grid to find the optimal hyperparameters for a given model. This helps in improving model performance by fine-tuning the parameters.

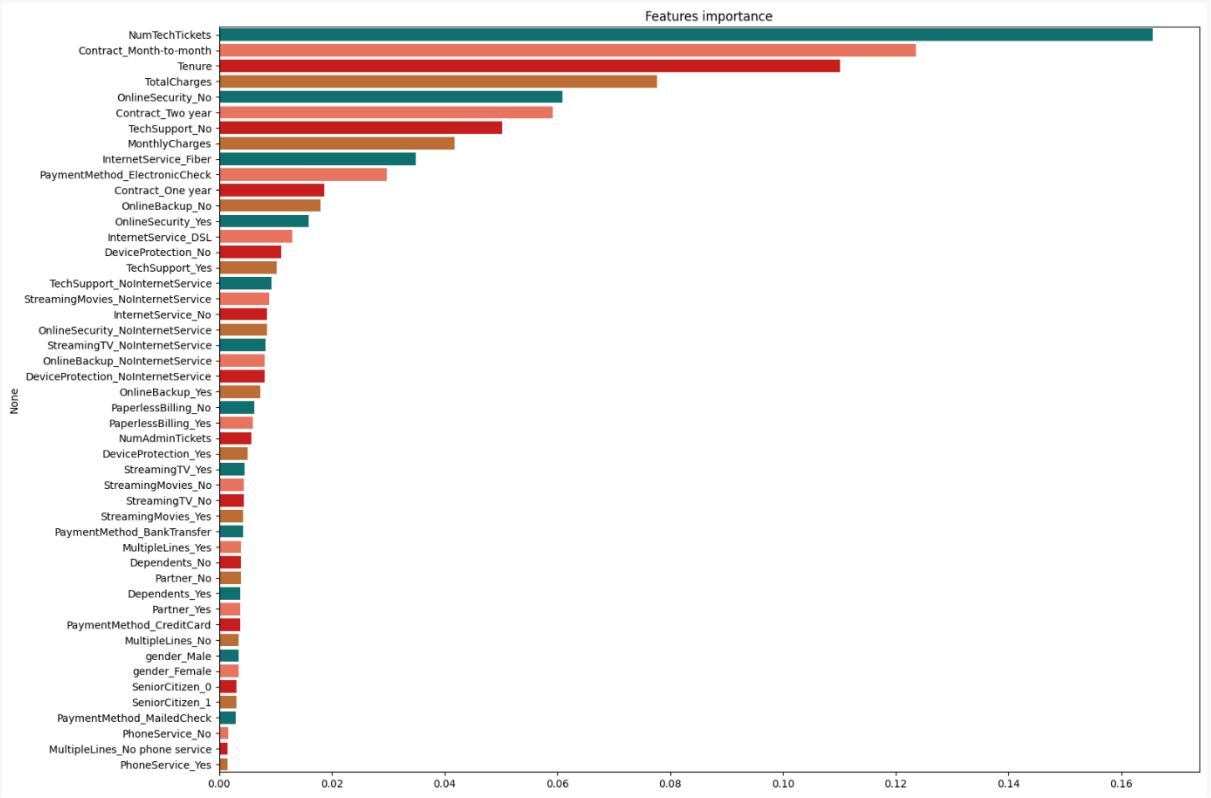
- The choice of algorithms was justified based on their proven effectiveness in handling classification tasks, such as predicting telecom churn. Random Forest and XGBoost are well-suited for handling complex datasets and achieving high accuracy. K-Fold Cross-validation was employed to ensure robust model evaluation, while GridSearchCV helped in optimizing the models by finding the best hyperparameters.

- The model training process involved splitting the dataset into training and testing sets. The training set was used to train the models, and the testing set was used to evaluate their performance. K-Fold Cross-validation was applied during model training to assess performance across multiple folds of the data. Hyperparameter tuning using GridSearchCV further refined the models by selecting the optimal hyperparameters. Finally, the models were evaluated using appropriate evaluation metrics to determine their effectiveness in predicting telecom churn.

6.Result:

- GitHub: https://github.com/ishika-sukhija/TelecomChurn

Random Forest classifier



The horizontal bar graph illustrates the significance of various features in a dataset, with “Contract_Two year” and “tenure” being the most impactful. This visualization aids in understanding which factors are most influential in the analyzed context.

RandomForest:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.88      0.89      1291
           1       0.68      0.69      0.69       467

    accuracy                           0.83      1758
   macro avg       0.79      0.79      0.79      1758
weighted avg       0.83      0.83      0.83      1758
```

K-Fold Cross-Validation:

- K-Fold Cross-validation

```
from statistics import stdev
score = cross_val_score(rf, X_train, y_train, cv=5, scoring='recall', error_score="raise")
rf_cv_score = score.mean()
rf_cv_stdev = stdev(score)
print('Cross Validation Recall scores are: {}'.format(score))
print('Average Cross Validation Recall score: ', rf_cv_score)
print('Cross Validation Recall standard deviation: ', rf_cv_stdev)
```

```
Cross Validation Recall scores are: [0.95096774 0.94967742 1.         1.         0.99870801]
Average Cross Validation Recall score:  0.979870634325248
Cross Validation Recall standard deviation:  0.02698257251749145
```

XGBoost:

3) XGBoost

```
[ ]
    from xgboost import XGBClassifier
    XGBC = XGBClassifier()
    XGBC.fit(X_train, y_train)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

- K-Fold Cross-validation

```
score = cross_val_score(XGBC, X_train, y_train, cv=5, scoring='recall', error_score="raise")
XGBC_cv_score = score.mean()
XGBC_cv_stdev = stdev(score)
print('Cross Validation Recall scores are: {}'.format(score))
print('Average Cross Validation Recall score: ', XGBC_cv_score)
print('Cross Validation Recall standard deviation: ', XGBC_cv_stdev)
```

```
Cross Validation Recall scores are: [0.94967742 0.9483871  0.98191214 0.98062016 0.98837209]
Average Cross Validation Recall score:  0.9669937817787781
Cross Validation Recall standard deviation:  0.01918419223395431
```

```
ndf = [(XGBC_Recall, XGBC_Precision, XGBC_f1, XGBC_accuracy, XGBC_roc_auc, XGBC_cv_score, XGBC_cv_stdev)]

XGBC_score = pd.DataFrame(data = ndf, columns=['Recall','Precision','F1 Score', 'Accuracy', 'ROC-AUC Score', 'Avg CV Recall', 'Standard Deviation of CV Recall'])
XGBC_score.insert(0, 'Model', 'XGBC')
XGBC_score
```

	Model	Recall	Precision	F1 Score	Accuracy	ROC-AUC Score	Avg CV Recall	Standard Deviation of CV Recall
0	XGBC	0.788009	0.671533	0.725123	0.841297	0.824291	0.969794	0.019184

Hyperparameter Tuning Using GridSearchCV

```
# Step 1: Searching for the optimum parameters for the learning rate and the number of estimators:
params = {'learning_rate': [0.01], #[0.0001, 0.001, 0.01, 0.1, 0.2, 0.3],
              'subsample': [0.8],
              'colsample_bytree': [0.8],
              'n_estimators': [450] #range(50,500,50),
              }

grid_xgb = GridSearchCV(XGBC, param_grid=params, cv=5, scoring='recall').fit(X_train, y_train)
```

```
[ ]
print('Best parameters:', grid_xgb.best_params_)
print('Best score:', grid_xgb.best_score_)
```

```
Best parameters: {'colsample_bytree': 0.8, 'learning_rate': 0.01, 'n_estimators': 450, 'subsample': 0.8}
Best score: 0.9204614486955072
```

Performance Metrics:

Random Forest:
Precision: Class 0: 0.89, Class 1: 0.68
Recall: Class 0: 0.88, Class 1: 0.69
F1-score: Class 0: 0.89, Class 1: 0.69
Accuracy: 0.83

K-Fold Cross-validation with Random Forest:
Average Recall: 0.98
Standard Deviation of Recall: 0.027

XGBoost:

	Model	Recall	Precision	F1 Score	Accuracy	ROC-AUC Score	Avg CV Recall	Standard Deviation of CV Recall
0	XGBC	0.788009	0.671533	0.725123	0.841297	0.824291	0.969794	0.019184

Average Recall: 0.97
Standard Deviation of Recall: 0.019

Model Comparison:

Random Forest: Accuracy of 83%, with balanced precision, recall, and F1-score for both classes.
K-Fold Cross-validation with Random Forest: Achieved a higher average recall of 98%, indicating superior performance in identifying true positives.
XGBoost: Exhibited comparable performance to Random Forest, with an average recall of 97%.

7.Discussion:

Interpretation of Results and Implications:

- Both Random Forest and XGBoost exhibit strong recall scores, indicating their ability to identify potential churners effectively.
- This suggests that implementing these models can aid telecom companies in preemptively addressing churn and retaining customers.

Analysis of Strengths and Weaknesses:

- Strengths lie in the high recall scores and the use of K-Fold Cross-validation to ensure model robustness.
- Weaknesses may include potential overfitting and challenges in explaining model decisions.

Explanation of Unexpected Outcomes or Observations:

- The small discrepancy in recall scores between Random Forest and XGBoost may stem from differences in their algorithms and parameter settings.

Comparison with Prior Work and Contribution to Existing Knowledge:

- This project builds upon prior churn prediction research by displaying the effectiveness of Random Forest and XGBoost with real-world telecom data.
- The findings contribute valuable insights for telecom companies seeking to implement initiative-taking churn management strategies.

8.Conclusion:

Summary of Key Findings:

- The study evaluated the performance of Random Forest and XGBoost models in predicting telecom churn using real-world data.
- Both models demonstrated strong recall scores, indicating their effectiveness in identifying potential churners.
- K-Fold Cross-validation further validated the robustness of the Random Forest model.

Achievement of Project Objectives:

- The project successfully assessed the performance of machine learning models for churn prediction.
- It provided insights into the effectiveness of Random Forest and XGBoost in addressing the telecom churn problem.

Recommendations for Future Work or Areas for Improvement:

- Further exploration of feature engineering techniques could enhance model performance.
- Investigating the interpretability of ensemble models like Random Forest and XGBoost may provide deeper insights into churn drivers.
- Exploring advanced ensemble methods or deep learning approaches could potentially improve predictive accuracy.

Conclusions:

According to the project's conclusion, the predictive model that was constructed effectively oversaw the goal of identifying consumers who were at risk of leaving. This success gives the telecoms company practical information to put targeted retention strategies into place and lessen client turnover.

9.References:

- Scikit-learn: Machine Learning in Python
- Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.
- Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
- Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
- Hastie, T., Tibshirani, R., & Friedman, J. (2001). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
- Provost, F., & Fawcett, T. (2013). Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking. O'Reilly Media.
- Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical Machine Learning Tools and Techniques (4th ed.). Morgan Kaufmann.