Churn Prediction Model Report

My Telecommunication Churn Prediction Model aims to address the crucial need for accurate forecasting of customer churn in the telecommunications industry through machine learning techniques like the **Categorical Boost Classifier** and **Random Forest Classifier**. The objective is to create a robust model capable of precisely identifying customers at risk of churn based on features such as **'Age'**, **'Martial Status'**, **'Monthly Charges'**, **'Gender',** and more. Anticipating customer churn is important for telecom companies, and my project involves comprehensive data collection, preprocessing, insightful feature extraction, and rigorous model selection.

TABLE OF CONTENTS

Explanation of the Model Used

Model Selection

I tried the model to work on two machine learning models:

1. Categorical Boost with SMOTETomek*(Higher Accuracy and F1 Score)*
2. Random Forest Classifier

Both models were trained on a dataset containing various customer features(**‘Age’**, **‘Martial Status’**, **‘Monthly Charges’**, **‘Gender’)** and with the Kaggle dataset provided in the Google Form. Here is the model Comparison for the Both -

Model Comparison

The Random Forest Classifier was one of the models I tested. It works by combining many decision trees to predict whether customers might leave. In my tests, it showed good accuracy(0.82) and a fair F1 score, indicating it's decent at predicting customer churn in our dataset.

Using RandomForest Classifier

```
[13]: from sklearn.ensemble import RandomForestClassifier
model_rf=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=8)
model_rf.fit(x_train,y_train)
y_pred=model_rf.predict(x_test)
model_rf.score(x_test,y_test)
print(classification_report(y_test, y_pred, labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.82      0.93      0.87      1023
           1       0.72      0.46      0.56       386

    accuracy                           0.80      1409
   macro avg       0.77      0.70      0.72      1409
weighted avg       0.79      0.80      0.79      1409
```

Accuracy achieved - 82%

F1 Score - 0.79

Observations: While Random Forest showed acceptable performance, it fell short in addressing class imbalance issues inherent in the dataset.

During my evaluation, I tried Categorical Boost with SMOTETomek. Categorical Boost builds models step by step, focusing on areas where previous models had trouble. This approach is great for fixing mistakes and works well with imbalanced data like in churn prediction. SMOTETomek, the trick I added, balances out the uneven data by adjusting how many examples it uses from each group. This combo helped tackle the imbalance issue we often face in churn prediction.

using Catboost with SMOTETomek

•[15]:

```
from imblearn.combine import SMOTETomek
from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

# Applying SMOTETomek here
smt = SMOTETomek(random_state=42)
X_resampled, y_resampled = smt.fit_resample(x, y)

# Splitting the resampled data
xr_train, xr_test, yr_train, yr_test = train_test_split(X_resampled, y_resampled, test_size=0.2)

# Initialize and train CatBoost model
model_catboost_smote = CatBoostClassifier(iterations=100, depth=6, learning_rate=0.1, random_state=100)
model_catboost_smote.fit(xr_train, yr_train)

# Predict on the test set
yr_predict = model_catboost_smote.predict(xr_test)

# Calculate model accuracy and other metrics
model_score_catboost = model_catboost_smote.score(xr_test, yr_test)
print(model_score_catboost)
print(metrics.classification_report(yr_test, yr_predict))
```

```
92:     learn: 0.2384443        total: 450ms    remaining: 33.9ms
93:     learn: 0.2378695        total: 454ms    remaining: 29ms
94:     learn: 0.2375820        total: 457ms    remaining: 24.1ms
95:     learn: 0.2372710        total: 461ms    remaining: 19.2ms
96:     learn: 0.2368010        total: 464ms    remaining: 14.4ms
97:     learn: 0.2361847        total: 468ms    remaining: 9.55ms
98:     learn: 0.2357539        total: 471ms    remaining: 4.76ms
99:     learn: 0.2351305        total: 475ms    remaining: 0us
0.868366285119667
              precision    recall  f1-score   support

           0       0.86      0.87      0.87       938
           1       0.87      0.87      0.87       984

    accuracy                           0.87      1922
   macro avg       0.87      0.87      0.87      1922
weighted avg       0.87      0.87      0.87      1922
```

Working of CatBoost Classifier Model

First, I used a technique called SMOTETomek to balance the dataset, making sure both groups (customers who left and those who stayed) had a fair representation.

Then, I split this adjusted data into two parts: one for teaching the model and the other to test how well it learned. Next, I Called the CatBoost library. This model was trained using the data meant for teaching. Once trained, the model predicted whether customers would churn or not based on their information.

Finally, I checked how accurate the model was by comparing its predictions to the actual results. This process helped me see how well the model could tell apart customers who churned from those who didn't.

Accuracy achieved - 86%

F1 Score - 0.87

Observations: Categorical Boost, particularly when coupled with SMOTETomek for handling imbalanced data, outperformed Random Forest in handling the class imbalance and predicting churn accurately.

More About SMOTETomek

SMOTETomek is a hybrid sampling method combining two techniques: SMOTE (Synthetic Minority Over-sampling Technique) and Tomek links. SMOTE generates synthetic samples for the minority class, while Tomek links identify and remove overlapping instances from different classes.

Why I used SMOTETomek

I used SMOTETomek to address the imbalanced dataset in my model. There are **fewer churners than Non-churners in the dataset(The ratio is 77:23)**, so to avoid overfitting I used it. By oversampling the minority class and undersampling the majority class simultaneously, it balances the dataset, making the model more robust in handling imbalanced data. This approach helps prevent the model from being biased towards the majority class and enhances its predictive performance for both classes.

EDA Findings

Below are the Findings gathered from EDA -

1. **Handling Missing Values using Interpolation**

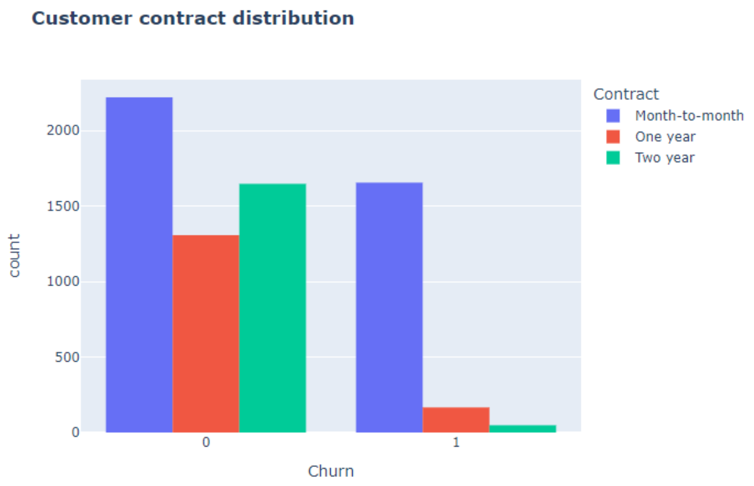
   At first, I focused on identifying any missing values, but the dataset had no musings values as such, except for a specific column **MonthlyCharges**. This column was initially in a different format and was converted to a numerical type, resulting in 11 missing values.

   To handle this, I employed a method called interpolation, which allowed me to estimate and fill in those missing values based on the surrounding data points. This approach ensured a complete dataset, enabling a more comprehensive analysis without losing any crucial information.

2. **Month to Month Contract customers are more Churnerns compared to one-year and Two-year**

   In my analysis, I created histogram to compare different contract types with the churn column. What I observed is that customers with monthly contracts tend to have a higher churn rate compared to other contract types.

```
[41]: fig = px.histogram(teleco, x="Churn", color = "Contract", barmode = "group", title = "<b>Customer contract distribution<b>")
      fig.update_layout(width=700, height=500, bargap=0.2)
      fig.show()
```



Observations: Month to Month Churn is more compared to one year and Two year

3. **SeniorCitizen who are using NO InternetService are loyal custmoers (Not Churn) with contract of Two-years paying more than 25 Monthly**

   Used the Groupby method to find that Senior Citizens who are using no Internet Service are also paying more and have a contract of Two-years and more. They are also paying more - 25 Monthly.

   They are Loyal Customers!

   Also, I grouped the dataset into different parameters to arrive at this conclusion.

```
[42]: teleco.groupby(['SeniorCitizen', 'InternetService', 'Contract' ,'MonthlyCharges', 'PaymentMethod','tenure','PaperlessBilling','Churn']).size().reset_inde
```

[42]:

	SeniorCitizen	InternetService	Contract	MonthlyCharges	PaymentMethod	tenure	PaperlessBilling	Churn	Count
0	0	DSL	Month-to-month	23.45	Electronic check	1	Yes	1	1
1	0	DSL	Month-to-month	23.90	Electronic check	13	Yes	1	1
2	0	DSL	Month-to-month	24.15	Electronic check	35	No	0	1
3	0	DSL	Month-to-month	24.20	Mailed check	1	No	0	1
4	0	DSL	Month-to-month	24.25	Electronic check	1	Yes	1	1
...	...	...	...	...	...	...	...	...	...
6916	1	No	Two year	25.10	Credit card (automatic)	72	Yes	0	1
6917	1	No	Two year	25.40	Bank transfer (automatic)	72	No	0	1
6918	1	No	Two year	25.45	Credit card (automatic)	71	No	0	1
6919	1	No	Two year	25.65	Credit card (automatic)	64	No	0	1
6920	1	No	Two year	25.70	Credit card (automatic)	72	Yes	0	1

6921 rows × 9 columns

Observations : SeniorCitizen who are using NO InternetService are loyal custmoers(Not Churn) with contract of Two-years paying more than 25 Monthly

4. Electronic check and Mailed check customers are more likely to Churn

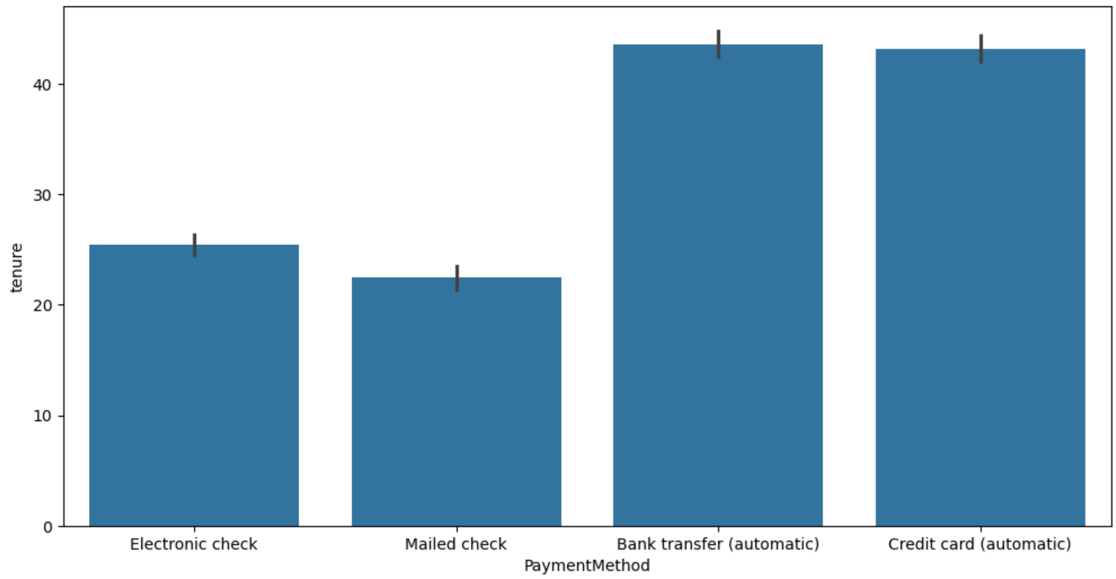
Electronic check and Mailed Check customers have a less tenure of purchase and are dependent on churn more likely.

```
[22]: import seaborn as sns
      import matplotlib
```

```
[23]: matplotlib.rcParams['figure.figsize'] = (12,6)
```

```
[24]: sns.barplot(x="PaymentMethod",y="tenure",data = temporary)
```

```
[24]: <Axes: xlabel='PaymentMethod', ylabel='tenure'>
```

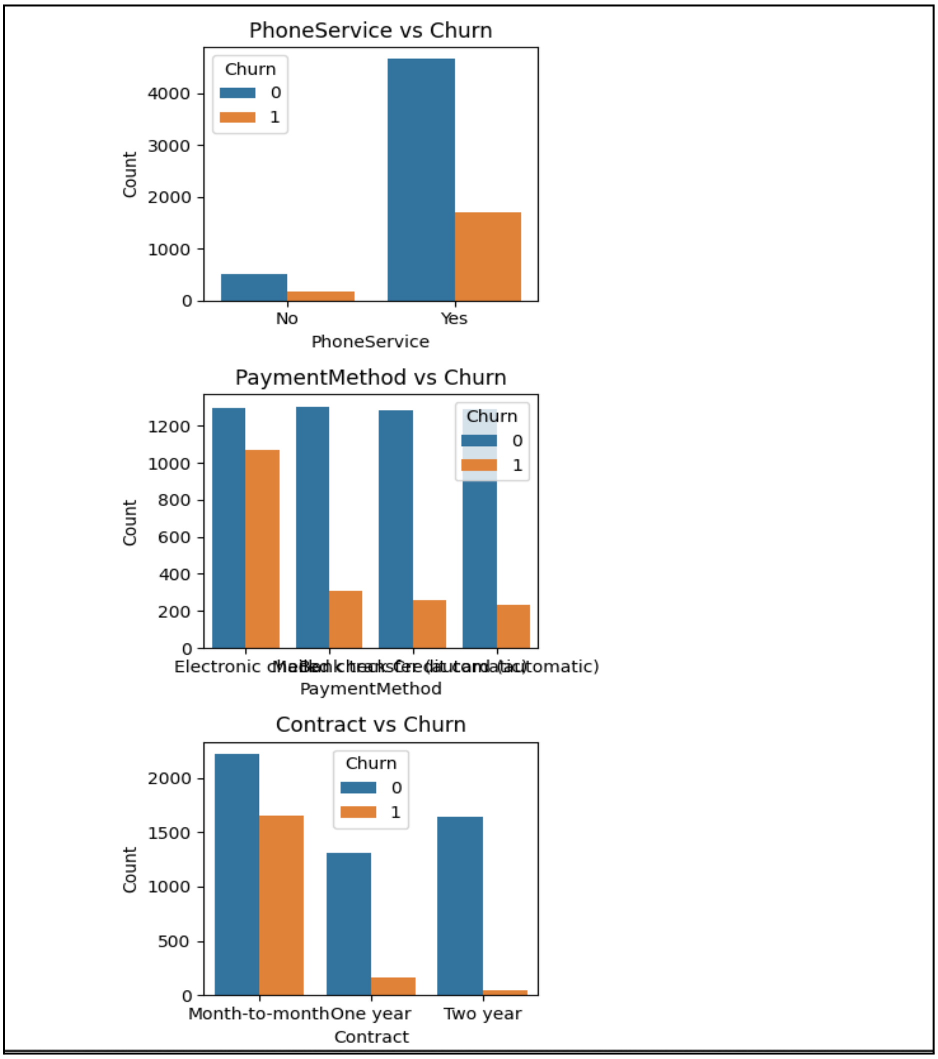


Electronic check and Mailed check customers are more likely to Churn

Feature Engineering

1. **Analyzed the Churn Feature/Column with other Features in a Bivariate form to find the correlation among them**

   I wanted to find out which things were most linked to customers deciding to leave. By looking at different details about customers, like how they pay or what services they use, I could see which parts were more connected to them leaving. This helped me pick out the most important details that could help us predict if a customer might churn in the future.



For instance, examining churn concerning factors like contract type, service usage, or customer demographics allowed us to identify potential patterns or dependencies.

2. **Grouping the different Features/Columns to gain insights for Precise prediction**

```
[43]: temporary = teleco.groupby(['SeniorCitizen', 'InternetService', 'Contract' ,'MonthlyCharges', 'PaymentMethod','tenure',
```

Grouping to check if SeniorCitizen or Non-SeniorCitizen are frequent churners or not. that Senior Citizens who are using no Internet Service are also paying more and have a contract of two years and more. They are also paying more - 25 Monthly. They are Loyal Customers!

3. **Creating Dummies and Plotting Heatmap to handle the categorical data and visualize the correlation matrix**

Created dummy data for the Categorical Data in the dataset to plot the correlation matrix in the form of a heatmap.

Generated a heatmap using the Seaborn library within a 12x12-inch figure. This heatmap visually represents the correlation between different features (columns) present in the *'teleco_dummies'*.

The heatmap color-codes these correlation values using a "Paired" colormap, where colors vary according to the strength and direction of correlation. Darker shades indicate a strong positive correlation between features, while lighter shades represent weaker correlations or no correlation at all. This visualization aids in identifying relationships among various features: positive correlations (near +1) imply that as one feature increases, the other tends to increase as well, whereas negative correlations (near -1) suggest an inverse relationship, where one feature increases as the other decreases.

[26]:
```
# Dropping the 'customerID' column
teleco.drop('customerID', axis=1, inplace=True)
```

[27]:
```
teleco_copy = teleco.copy()
```

[28]:
```
teleco_copydemo = pd.get_dummies(teleco_copy, dtype=int)

teleco_copydemo.head()
```

[28]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	...	StreamingMovies_Yes	Conti
0	0	1	29.85	29.85	0	1	0	0	1	1	...	0	
1	0	34	56.95	1889.50	0	0	1	1	0	1	...	0	
2	0	2	53.85	108.15	1	0	1	1	0	1	...	0	
3	0	45	42.30	1840.75	0	0	1	1	0	1	...	0	
4	0	2	70.70	151.65	1	1	0	1	0	1	...	0	

5 rows × 46 columns

[29]:
```
plt.figure(figsize=(24, 22))
sns.heatmap(teleco_copydemo.corr(), annot=True, fmt=".2f",  linewidths=1)
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```

Evaluation Result

Results with CatBoost Classifier

1. With an overall accuracy of 87%, the model demonstrates correct in predicting both churn (1) and non-churn (0) instances within the dataset.

2. Precision and recall, measured at 87% for both classes, indicate the model's ability to accurately classify positive (churn) and negative (non-churn) instances, respectively.

3. The F1-score, a mean of precision and recall, stands at 0.87 for both churn and non-churn classes, suggesting a balance between precision and recall for both categories.

4. The support column indicates the number of actual occurrences for each class within the dataset.

5. This model achieves consistency in its performance across both classes, showing a reliable predictive ability and balanced performance between identifying churn and non-churn cases.

```
92:	learn: 0.2384443	total: 450ms	remaining: 33.9ms
93:	learn: 0.2378695	total: 454ms	remaining: 29ms
94:	learn: 0.2375820	total: 457ms	remaining: 24.1ms
95:	learn: 0.2372710	total: 461ms	remaining: 19.2ms
96:	learn: 0.2368010	total: 464ms	remaining: 14.4ms
97:	learn: 0.2361847	total: 468ms	remaining: 9.55ms
98:	learn: 0.2357539	total: 471ms	remaining: 4.76ms
99:	learn: 0.2351305	total: 475ms	remaining: 0us
0.868366285119667
              precision    recall  f1-score   support

           0       0.86      0.87      0.87       938
           1       0.87      0.87      0.87       984

    accuracy                           0.87      1922
   macro avg       0.87      0.87      0.87      1922
weighted avg       0.87      0.87      0.87      1922
```

Challenges Faced

1. **Imbalanced Dataset**

   The dataset was imbalance between churn and non-churn instances, which affects the model's ability to learn patterns effectively. With a 77:23 ratio of imbalance, it was overfitted on the non-churners and that affected the prediction.

   I used the upsampling method to avoid this. Using Catboost with the tomek links did the work for me to avoid overfitting.

2. **Non-Correlated Features**

   Identifying highly correlated features required careful analysis to avoid issues. Also, many columns were not even correlating with each other - like Male-Female to the churn. They had a 50:50 dependency.

3. **Handling Categorical data values**

   Converted the categorical values into the numerical values using the dummies method so as to plot them in many plots and heatmap. Categorical variables contained missing values, which need to be handled appropriately. Used interpolation for the same.