Telco Customer Churn Prediction

Naomi Anastasya Simatupang

Digital Skola - Data Science

Link Project : https://github.com/naomiachoo/Telco-Customer-Churn-Project

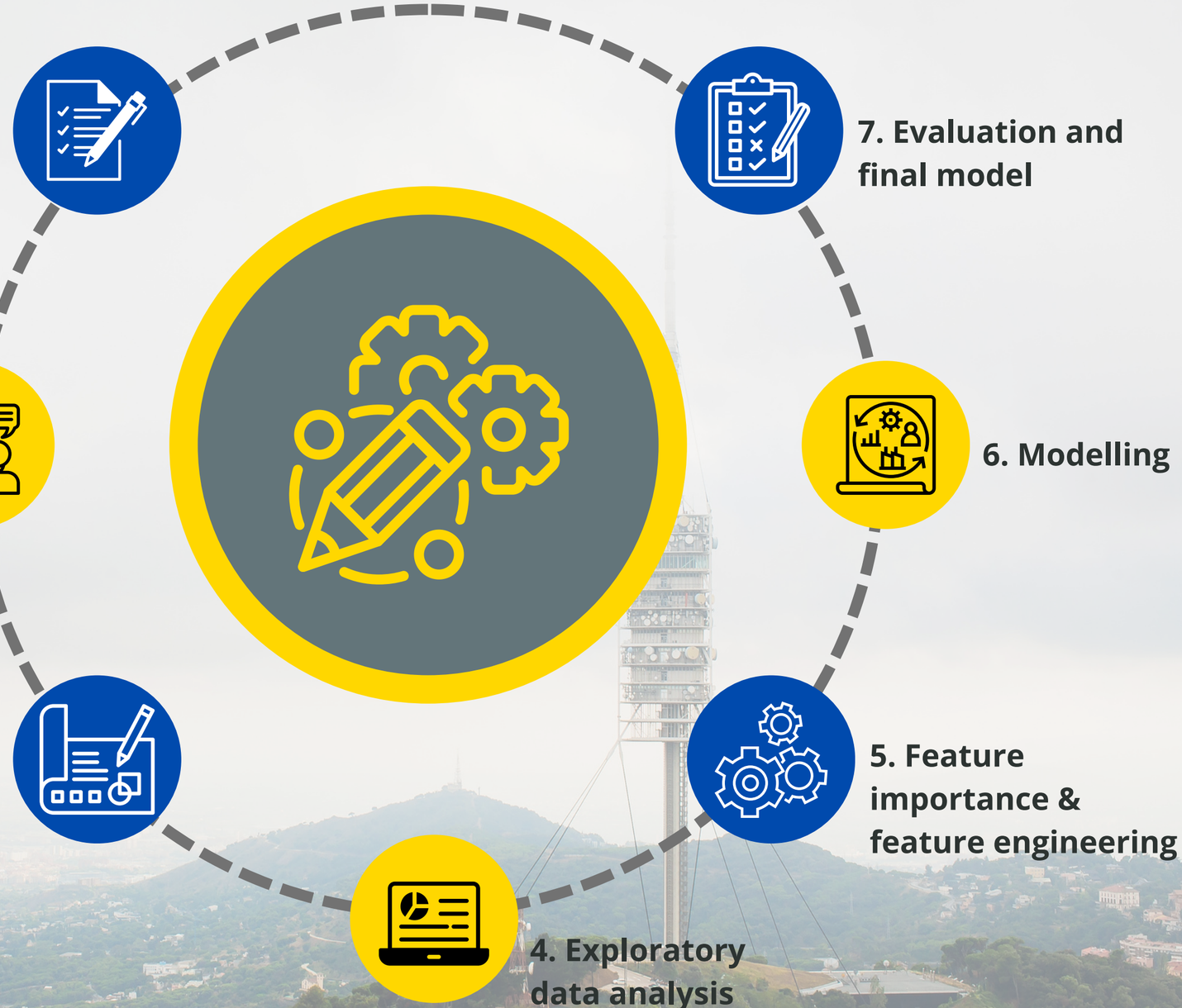
AGENDA

1. Background
2. Identification of Activities
3. Understanding Business & Data
4. Specification of Data
5. Data Preparation & Pre-Processing
6. Exploratory Data Analysis
7. Feature Importance
8. Feature Engineering
9. Modelling
10. Evaluation & Conclussion

BACKGROUND

- Dataset telco customer churn berisi detail dari data pelanggan yaitu "customerI ID", "gender", "payment method", "monthly-total charge" dan "status". Status dari customer churn mendefinisikan pelanggan mana yang pergi dan tidak pergi.
- Tujuan dari project ini adalah untuk membuat sebuah model machine learning yang dapat melakukan prediksi pelanggan mana yang akan churn atau tidak

IDENTIFICATION OF ACTIVITIES



UNDERSTANDING BUSINESS & DATA

- Customer churn merupakan jumlah dari hilangnya pelanggan yang menggunakan layanan/jasa/produk dari perusahaan dengan berbagai alasan. Terdapat beberapa keuntungan apabila melakukan analisis mengenai customer churn seperti kesempatan untuk meningkatkan keuntungan, meningkatkan kepuasan pelanggan, mengetahui target pasar dan meningkatkan kualitas dari produk.

UNDERSTANDING BUSINESS & DATA

- Memahami definisi dari setiap kolom pada dataset. Pada data ini terdapat beberapa kolom dengan informasi sebagai berikut :
  - Gender : Informasi jenis kelamin pelanggan apakah pria atau wanita
  - Payment method : Informasi jenis pembayaran dari pelanggan
  - Monthly charges : Tagihan bulanan dari pelanggan
  - Total charges : Jumlah tagihan pelanggan selama berlangganan
- Memahami tipe data dari setiap kolom seperti kategorikal dan numerikal
- Memahami isi dari setiap kolom pada data
  - Gender : Female/Male
  - Payment Method : Bank Transfer, Credit Card, Electronic Check, Mailed Check
  - Monthly Charges : Berisi nilai pembayaran pelanggan setiap bulan
  - Total Charges : Berisi nilai total pembayaran pelanggan
  - Churn : Yes/No

SPECIFICATION OF DATA

Column Name	Description	Data Type
Gender	jenis kelamin pelanggan	Kategorical
Payment Method	metodde pembayaran pelanggan	Kategorical
Monthly Charge	jumlah pembayaran bulanan pelanggan	Numeric
Total Charge	jumlah pembayaran keseluruhan pelanggan	Numeric
status	status churn atau tidak pelanggan	Kategorical

DATA PREPARATION & PRE-PROCESSING

IMPORT LIBRARY

```
#Import library
import numpy as np
import pandas as pd

#Import visualization library
import matplotlib.pyplot as plt
import seaborn as sns

#import math
import math

from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import classification_report, accuracy_score, f1_score, roc_curve, auc, roc_auc_score
from sklearn.metrics import mutual_info_score
from sklearn import metrics


import warnings
warnings.filterwarnings('ignore')
```

DATA READING

Data ini memiliki 6 independen variabel sebagai berikut :

1. customerID : Berisi informasi berupa nilai unik, dimana setiap pelanggan memiliki nilai customerID yang berbeda
2. gender : Berisi informasi jenis kelamin pelanggan, apakah female atau male
3. PaymentMethod : Berisi informasi metode pembayaran pelanggan, terdapat 4 nilai unik yaitu Electronic check, Mailed check, Bank transfer (automatic), Credit Card (automatic)
4. MontlyCharges : Berisi nilai pembayaran pelanggan setiap bulan
5. TotalCharges : Berisi nilai total pembayaran pelanggan
6. Churn : Berisi status Yes/No

```
[2] telco_data = pd.read_csv('https://raw.githubusercontent.com/naomiachoo/data-science-project-one-voice/mai
    telco_data.head()
```

	customerID	gender	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	7590-VHVEG	Female	Electronic check	29.85	29.85	No
1	5575-GNVDE	Male	Mailed check	56.95	1889.5	No
2	3668-QPYBK	Male	Mailed check	53.85	108.15	Yes
3	7795-CFOCW	Male	Bank transfer (automatic)	42.30	1840.75	No
4	9237-HQITU	Female	Electronic check	70.70	151.65	Yes

CHECK COLUMNS DATA TYPE

```
[3]  #check columns data type
     data_type = pd.DataFrame(telco_data.dtypes).T.rename(index={0:'Columns Type'})
     data_type
```

	customerID	gender	PaymentMethod	MonthlyCharges	TotalCharges	Churn
Columns Type	object	object	object	float64	object	object

Setelah dilakukan pengecekan terhadap tipe data dari setiap kolom, didapatkan bahwa kolom TotalCharges bertipe data object. Kolom ini berisi angka-angka desimal, tentunya tipe data yang sesuai adalah float.

CHECK COLUMNS DATA TYPE

```
[4]  #function to replace number separator
     def replacee(s):
         i=str(s).find(',')
         if(i>0):
             return s[:i] + '.' + s[i+1:]
         else :
             return s
```

Sebelum dilakukan konversi dari object ke float, perlu diperiksa apakah terdapat value yang memiliki separator "," Setelah diperiksa, apabila ada maka separator tersebut harus diubah ke "." untuk dapat dilakukan konversi ke tipe data float. Penjelasan ini adalah untuk fungsi `replacee`

CHECK COLUMNS DATA TYPE

```
[5]  #change the number separator
     telco_data['TotalCharges'] = telco_data['TotalCharges'].apply(replacee)
```

```
[6]  #convert TotalCharges dtype
     telco_data['TotalCharges'] = pd.to_numeric(telco_data['TotalCharges'], errors = 'coerce')
     print(telco_data['TotalCharges'].dtypes)
```

```
float64
```

```
[44] #check columns data type
     data_type = pd.DataFrame(telco_data.dtypes).T.rename(index={0:'Columns Type'})
     data_type
```

	gender	PaymentMethod	MonthlyCharges	TotalCharges	Churn
Columns Type	object	object	float64	float64	object

CHECK NULL VALUES

```
#check null values
null_val = data_type.append(pd.DataFrame(telco_data.isnull().sum()).T.rename(index = {0:'Amount of Null Values'}))
null_val
```

	customerID	gender	PaymentMethod	MonthlyCharges	TotalCharges	Churn
Columns Type	object	object	object	float64	float64	object
Amount of Null Values	0	0	0	0	11	0

```
#check null values
null_val = null_val.append(pd.DataFrame(round(telco_data.isnull().sum()/telco_data.shape[0]*100,2)).T.rename(index={0:'Percentage null values'}))
null_val.T
```

	Columns Type	Amount of Null Values	Percentage null values
customerID	object	0	0.0
gender	object	0	0.0
PaymentMethod	object	0	0.0
MonthlyCharges	float64	0	0.0
TotalCharges	float64	11	0.16
Churn	object	0	0.0

CHECK ANY MISSING VALUES

```python
def checking_null_values(dataset):
    """
    show null values and its percentage
    """
    print('Dimension of the dataset', dataset.shape)
    null_val = pd.DataFrame(dataset.dtypes).T.rename(index={0:'Columns Type'})
    null_val = null_val.append(pd.DataFrame(dataset.isnull().sum()).T.rename(index = {0:'Amount of Null Values'}))
    null_val = null_val.append(pd.DataFrame(round(dataset.isnull().sum()/dataset.shape[0]*100,2)).T.rename(index={0:'Percentage null values'}))
    return null_val.T
```

```python
#exclude the unnamed variable
data = telco_data.iloc[:, 1:]

#show null values and its percentage
checking_null_values(data)
```

```
Dimension of the dataset (7043, 5)
```

	Columns Type	Amount of Null Values	Percentage null values
gender	object	0	0.0
PaymentMethod	object	0	0.0
MonthlyCharges	float64	0	0.0
TotalCharges	float64	11	0.16
Churn	object	0	0.0

Dilakukan pengecekan pada data dan terdapat 11 nilai null pada kolom TotalCharges

PAYMENT METHOD DENOMINATION

```
[11] #unique element of PaymentMethod
     telco_data.PaymentMethod.unique()
```

```
array(['Electronic check', 'Mailed check', 'Bank transfer (automatic)',
       'Credit card (automatic)'], dtype=object)
```

Pada kolom PaymentMethod, terdapat value yang memiliki kata (automatic), kata ini terlalu panjang apabila dilakukan visualisasi nanti, oleh sebab itu akan dihilangkan.

```
[12] #remove (automatic) from payment method
     telco_data['PaymentMethod'] = telco_data['PaymentMethod'].str.replace(' (automatic)', '', regex=False)
```

REMOVE CUSTOMER ID COLUMN

```
[13] telco_data.drop(columns = 'customerID', inplace = True)
```

customerID tidak diperlukan karena tidak menjelaskan apakah pelanggan akan churn atau tidak.

DROP NULL VALUES

```
telco_data[telco_data['TotalCharges'].isnull()]
```

	gender	PaymentMethod	MonthlyCharges	TotalCharges	Churn
488	Female	Bank transfer	52.55	NaN	No
753	Male	Mailed check	20.25	NaN	No
936	Female	Mailed check	80.85	NaN	No
1082	Male	Mailed check	25.75	NaN	No
1340	Female	Credit card	56.05	NaN	No
3331	Male	Mailed check	19.85	NaN	No
3826	Male	Mailed check	25.35	NaN	No
4380	Female	Mailed check	20.00	NaN	No
5218	Male	Mailed check	19.70	NaN	No
6670	Female	Mailed check	73.35	NaN	No
6754	Male	Bank transfer	61.90	NaN	No

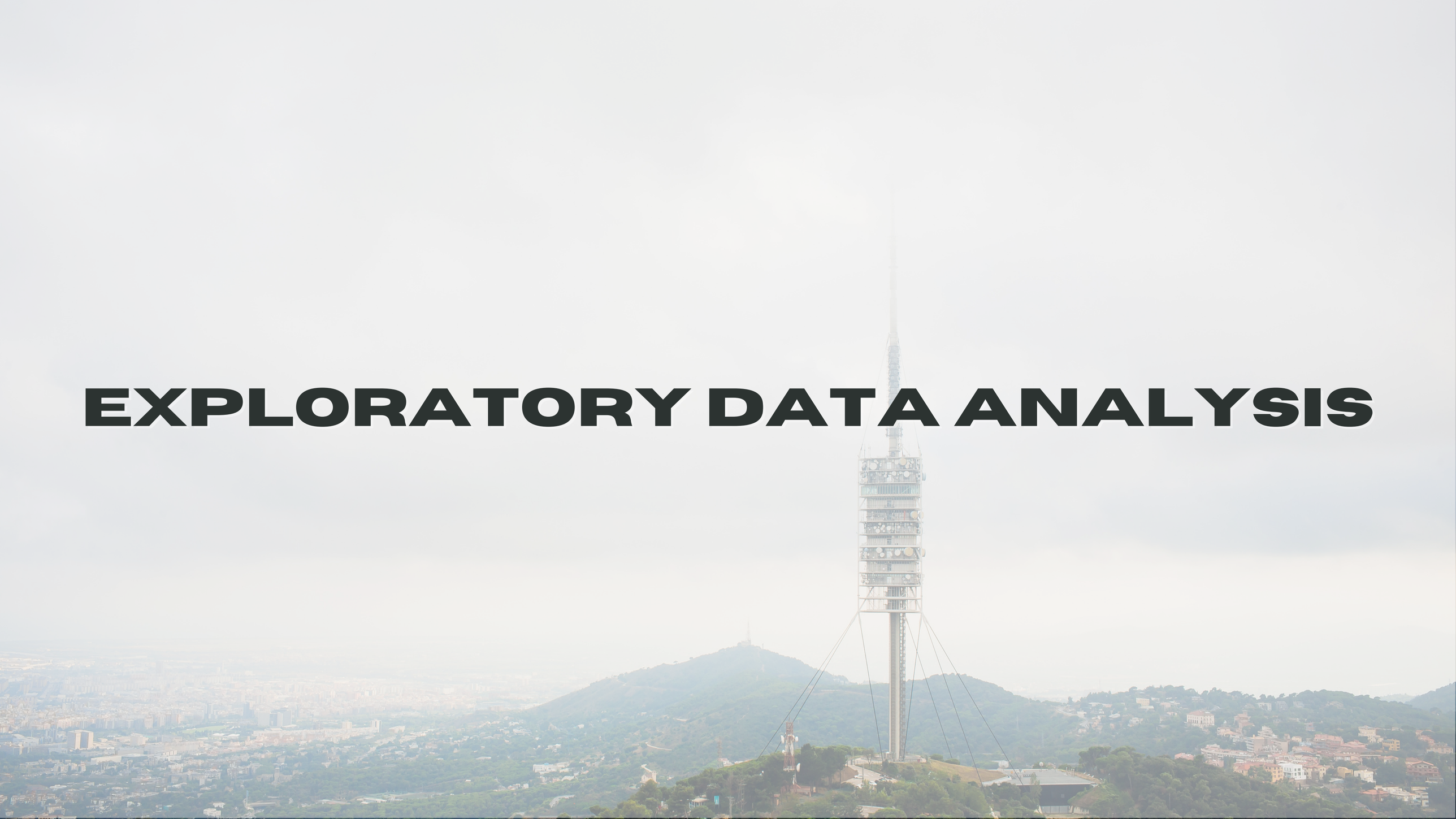
```
[15] # drop null values
telco_data = telco_data.dropna()

#show null values and its percentage
checking_null_values(telco_data)
```

```
Dimension of the dataset (7032, 5)
```

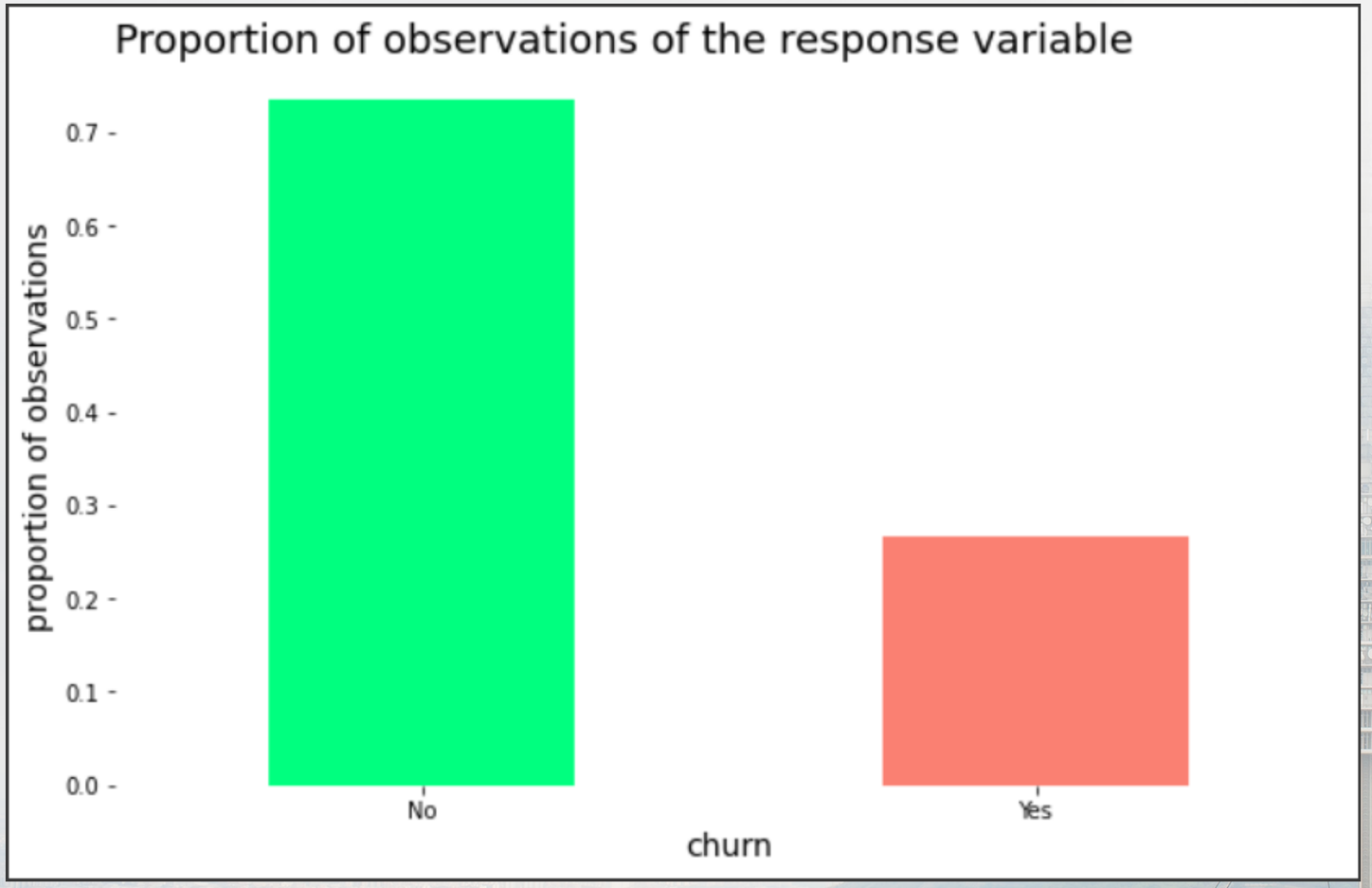
	Columns Type	Amount of Null Values	Percentage null values
gender	object	0	0.0
PaymentMethod	object	0	0.0
MonthlyCharges	float64	0	0.0
TotalCharges	float64	0	0.0
Churn	object	0	0.0

Pada data terdapat 11 baris yang memiliki TotalCharges 0, tidak ada informasi lain dari data yang mendukung mengapa TotalCharges yang null harus diganti nilai nya atau harus dihapus, namun mempertahankan nilai null akan mempengaruhi dalam melakukan pemodelan, oleh sebab itu kami memutuskan untuk menghapus nilai null tersebut.

EXPLORATORY DATA ANALYSIS



DATA VISUALIZATION - RESPONSE VARIABLE



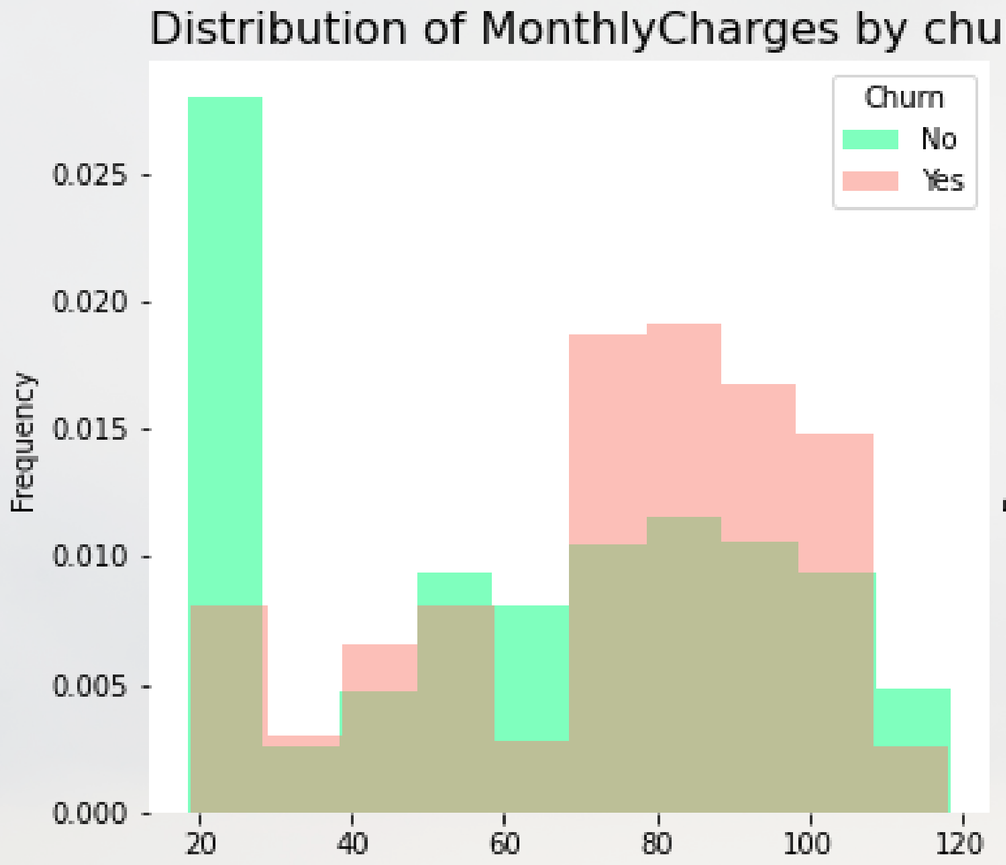

```
[17] #count totals of response variable
     df_response_total = pd.DataFrame(telco_data['Churn'].value_counts())

     #Count the percentage
     pd.set_option('display.float_format', '{:.2%}'.format)
     df_response_percentage = pd.DataFrame(telco_data['Churn'].value_counts(normalize=True))

     #concat the dataframe
     df_concats = pd.concat([df_response_total,df_response_percentage], axis=1, join="inner")
     df_concats.columns = ["Total", "Percentage"]

     df_concats
```

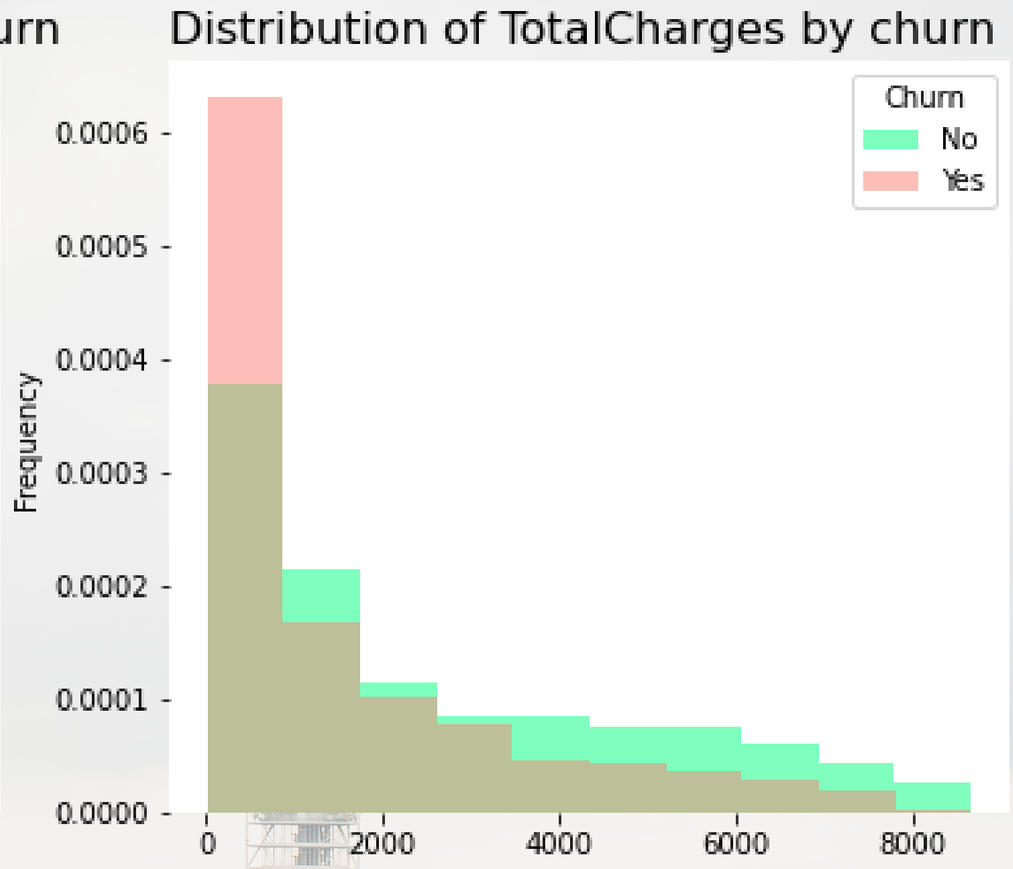
	Total	Percentage
No	5163	73.42%
Yes	1869	26.58%

Setelah melakukan observasi terhadap response variable kita mendapatkan bahwa terdapat customer yang churn dan tidak churn.

Mayoritas customer tidak melakukan churn dengan persentasi 73,46%, sedangkan yang churn adalah 26,54%

DATA VISUALIZATION - NUMERICAL VARIABLE

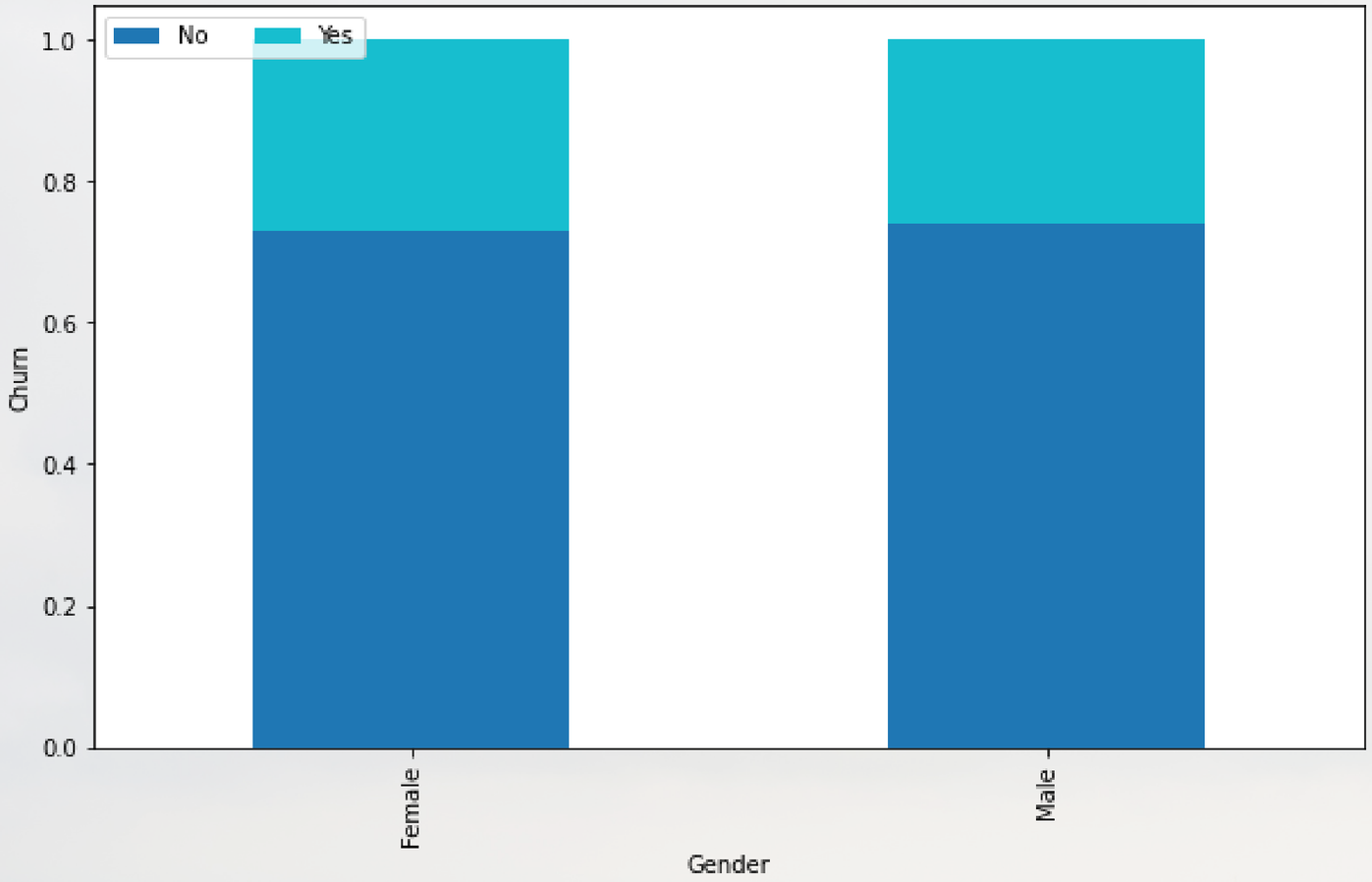




Berdasarkan analisis pada Monthly Charges dan Total Charges maka didapatkan beberapa kesimpulan yaitu :

- Semakin besar monthly charges maka probabilitas churn juga mayoritas akan semakin tinggi
- Customer dengan total charges yang tinggi kemungkinan untuk churn semakin rendah

PERCENTAGE OF CHURN FOR GENDER CATEGORY




```
cross_tab_prop_gender = pd.crosstab(index=data['gender'],
                                    columns=data['Churn'],
                                    normalize="index")
cross_tab_prop_gender
```

gender	No	Yes
Female	73.04%	26.96%
Male	73.80%	26.20%

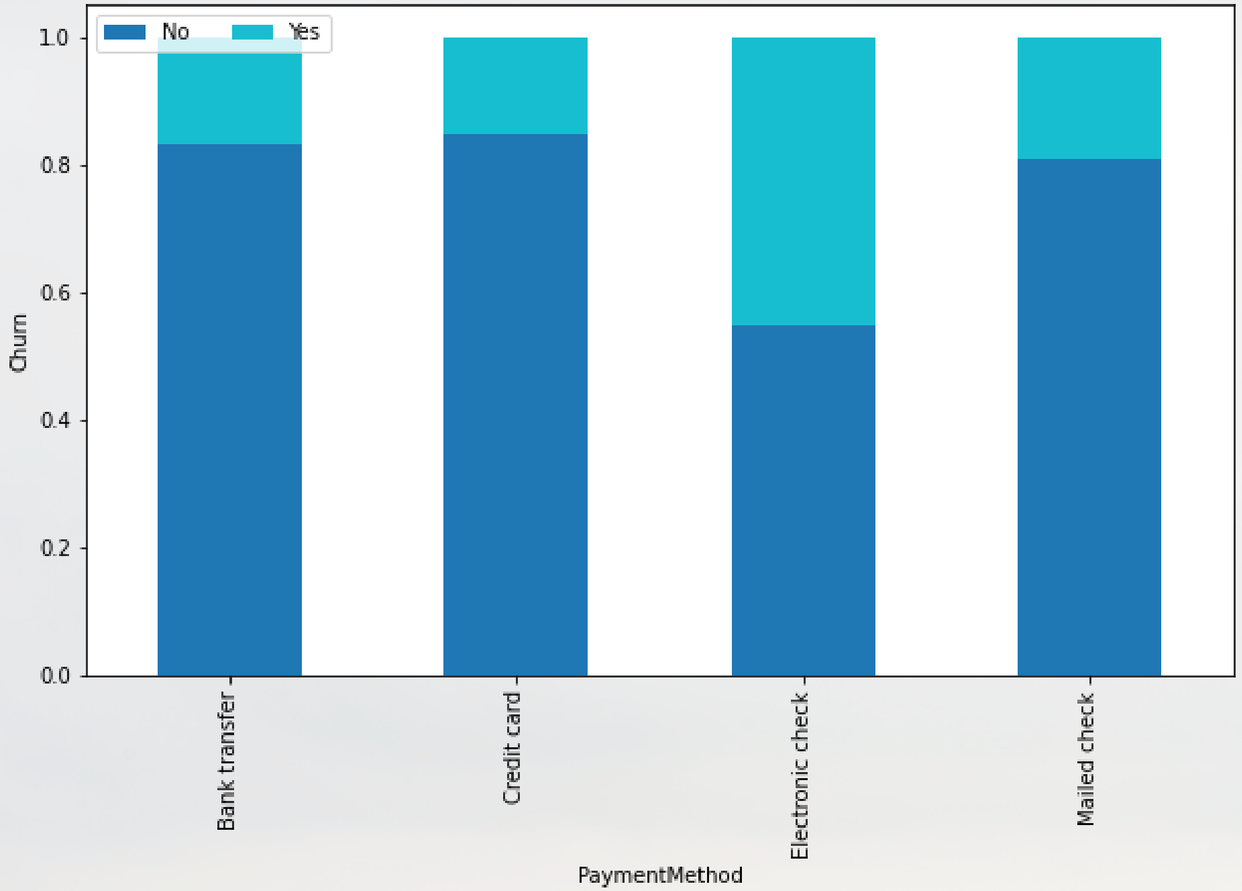
```
[21] cross_tab_gender = pd.crosstab(index=data['gender'],
                                    columns=data['Churn'])
     cross_tab_gender
```

gender	No	Yes
Female	2544	939
Male	2619	930

Berdasarkan analisis pada persentasi churn terhadap gender maka didapatkan beberapa kesimpulan yaitu :

- Baik perempuan dan laki-laki lebih dominan untuk tidak churn.
- Perempuan memiliki persentasi 73.04% untuk tidak churn
- Laki-laki memiliki persentasi 73.80% untuk tidak churn

PERCENTAGE OF CHURN FOR PAYMENT METHOD CATEGORY




```
cross_tab_prop_pm = pd.crosstab(index=data['PaymentMethod'],
                                columns=data['Churn'],
                                normalize="index")
cross_tab_prop_pm
```

PaymentMethod	No	Yes
Bank transfer	83.27%	16.73%
Credit card	84.75%	15.25%
Electronic check	54.71%	45.29%
Mailed check	80.80%	19.20%

```
cross_tab_pm = pd.crosstab(index=data['PaymentMethod'],
                           columns=data['Churn'])
cross_tab_pm
```

PaymentMethod	No	Yes
Bank transfer	1284	258
Credit card	1289	232
Electronic check	1294	1071
Mailed check	1296	308

Berdasarkan analisis pada persentasi churn terhadap payment method maka didapatkan beberapa kesimpulan yaitu :

- Seluruh payment method memiliki persentasi yang dominan untuk tidak churn
- Bank transfer memiliki persentasi 83,27% untuk tidak churn
- Credit card memiliki persentasi 84,75% untuk tidak churn
- Electronic check memiliki persentasi 54,71% untuk tidak churn
- Mailed check memiliki persentasi 80,80% untuk tidak churn

FEATURE IMPORTANCE

```python
# function that computes the mutual infomation score between a categorical serie and the column Churn
def compute_mutual_information(categorical_serie):
    return mutual_info_score(categorical_serie, telco_data.Churn)

# select categorial variables excluding the response variable
categorical_variables = telco_data.select_dtypes(include=object).drop('Churn', axis=1)

# compute the mutual information score between each categorical variable and the target
feature_importance = categorical_variables.apply(compute_mutual_information).sort_values(ascending=False)

# visualize feature importance
print(feature_importance)
```

```
PaymentMethod    4.44%
gender           0.00%
dtype: float64
```

Menggunakan Mutual Information selain membantu dalam memahami data, kita juga dapat mengidentifikasi variabel mana yang mempengaruhi target variabel yang ada pada data. Score yang dimiliki oleh variabel PaymentMethod dan gender, apabila semakin mendekati 0 maka dapat disimpulkan bahwa variabel tersebut tidak mempengaruhi variabel target. Oleh sebab itu kita dapat menyimpulkan bahwa variabel PaymentMethod yang memiliki score 4,45% mempengaruhi hasil pada variabel target.

FEATURE ENGINEERING

LABEL ENCODING

```python
telco_data_transformed = telco_data.copy()

# label encoding (binary variables)
label_encoding_columns = ['gender', 'Churn']

# encode categorical binary features using label encoding
for column in label_encoding_columns:
    if column == 'gender':
        telco_data_transformed[column] = telco_data_transformed[column].map({'Female': 1, 'Male': 0})
    else:
        telco_data_transformed[column] = telco_data_transformed[column].map({'Yes': 1, 'No': 0})
```

```python
telco_data_transformed.head()
```

	gender	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	1	Electronic check	2985.00%	2985.00%	0
1	0	Mailed check	5695.00%	188950.00%	0
2	0	Mailed check	5385.00%	10815.00%	1
3	0	Bank transfer	4230.00%	184075.00%	0
4	1	Electronic check	7070.00%	15165.00%	1

ONE HOT ENCODING

```
# one-hot encoding (categorical variables with more than two levels)
one_hot_encoding_columns = ['PaymentMethod']

# encode categorical variables with more than two levels using one-hot encoding
telco_data_transformed = pd.get_dummies(telco_data_transformed, columns = one_hot_encoding_columns)
```

```
telco_data_transformed.head()
```

	gender	MonthlyCharges	TotalCharges	Churn	PaymentMethod_Bank transfer	PaymentMethod_Credit card	PaymentMethod_Electronic check	PaymentMethod_Mailed check
0	1	2985.00%	2985.00%	0	0	0	1	0
1	0	5695.00%	188950.00%	0	0	0	0	1
2	0	5385.00%	10815.00%	1	0	0	0	1
3	0	4230.00%	184075.00%	0	1	0	0	0
4	1	7070.00%	15165.00%	1	0	0	1	0

one hot encoding menciptakan kolom binary baru untuk setiap level dari variable kategorikal. Berisi 0 dan 1 untuk mengindikasikan ada atau tidak nya kategori pada data. One hot encoding diterapkan pada PaymentMethod pada penelitian ini.

NORMALIZATION

```
# min-max normalization (numeric variables)
min_max_columns = ['MonthlyCharges', 'TotalCharges']

# scale numerical variables using min max scaler
for column in min_max_columns:
        # minimum value of the column
        min_column = telco_data_transformed[column].min()
        # maximum value of the column
        max_column = telco_data_transformed[column].max()
        # min max scaler
        telco_data_transformed[column] = (telco_data_transformed[column] - min_column) / (max_column - min_column)
```

```
[36] telco_data_transformed.head()
```

	gender	MonthlyCharges	TotalCharges	Churn	PaymentMethod_Bank transfer	PaymentMethod_Credit card	PaymentMethod_Electronic check	PaymentMethod_Mailed check
0	1	11.54%	0.13%	0	0	0	1	0
1	0	38.51%	21.59%	0	0	0	0	1
2	0	35.42%	1.03%	1	0	0	0	1
3	0	23.93%	21.02%	0	1	0	0	0
4	1	52.19%	1.53%	1	0	0	1	0

MODELLING

SPLITTING DATA TRAIN & DATA TEST

```
# select independent variables
X = telco_data_transformed.drop(columns='Churn')

# select dependent variables
y = telco_data_transformed.loc[:, 'Churn']

# prove that the variables were selected correctly
print(X.columns)

# prove that the variables were selected correctly
print(y.name)
```

```
Index(['gender', 'MonthlyCharges', 'TotalCharges',
       'PaymentMethod_Bank transfer', 'PaymentMethod_Credit card',
       'PaymentMethod_Electronic check', 'PaymentMethod_Mailed check'],
      dtype='object')
Churn
```

```
[39] #Perform train test split data
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)
```

FUNCTION FOR EVALUATION METRICS

```
[45] def evaluation_metrics(model):
        """
        Evaluation metrics for classification ML project
        """
        y_predict_train = model.predict(X_train)
        y_predict_test = model.predict(X_test)

        training_acc = accuracy_score(y_train, y_predict_train)
        testing_acc = accuracy_score(y_test, y_predict_test)

        print("Training Accuracy: {}".format(training_acc))
        print("Testing Accuracy: {}".format(testing_acc))

        print(classification_report(y_test, y_predict_test))
```

LOGISTIC REGRESSION

```python
# Logistic Regression
lr = LogisticRegression()
lr.fit(X_train, y_train) ## fill the code

evaluation_metrics(lr)
```

```
Training Accuracy: 0.7834666666666666
Testing Accuracy: 0.7867803837953091
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      1050
           1       0.61      0.45      0.51       357

    accuracy                           0.79      1407
   macro avg       0.72      0.67      0.69      1407
weighted avg       0.77      0.79      0.77      1407
```

DECISION TREE

```
[47] # Decision Tree
     dt = DecisionTreeClassifier()
     dt.fit(X_train, y_train) ## fill the code

     evaluation_metrics(dt)
```

```
     Training Accuracy: 0.9950222222222223
     Testing Accuracy: 0.720682302771855
                   precision    recall  f1-score   support

                0       0.82      0.80      0.81      1050
                1       0.45      0.49      0.47       357

         accuracy                           0.72      1407
        macro avg       0.64      0.64      0.64      1407
     weighted avg       0.73      0.72      0.72      1407
```

GRADIENT BOOSTING

```
# Gradient Boosting
gb = GradientBoostingClassifier()
gb.fit(X_train, y_train) ## fill the code

evaluation_metrics(gb)
```

```
Training Accuracy: 0.8151111111111111
Testing Accuracy: 0.7803837953091685
              precision    recall  f1-score   support

           0       0.84      0.87      0.86      1050
           1       0.58      0.50      0.54       357

    accuracy                           0.78      1407
   macro avg       0.71      0.69      0.70      1407
weighted avg       0.77      0.78      0.78      1407
```

EVALUATION & CONCLUSION

EVALUATION

Description	Precision	Recall	f1-Score	Accuracy
Logistic Regresion	**0.83**	**0.90**	**0.86**	**0.79**
Decision Tree	**0.83**	**0.8**	**0.81**	**0.72**
Gradient Boosting	**0.84**	**0.87**	**0.86**	**0.78**

CONCLUSION

Project ini menghasilkan sebuah kesimpulan bahwa model yang terbaik untuk digunakan adalah Gradient Boosting hal ini dikarenakan pada project ini kami menitik beratkan hasil evaluasi pada precission, dimana data pelanggan yang benar benar churn lah yang ingin dilihat bagaimana pola nya dan membantu perusahaan dalam mengambil keputusan.


THANK
YOU