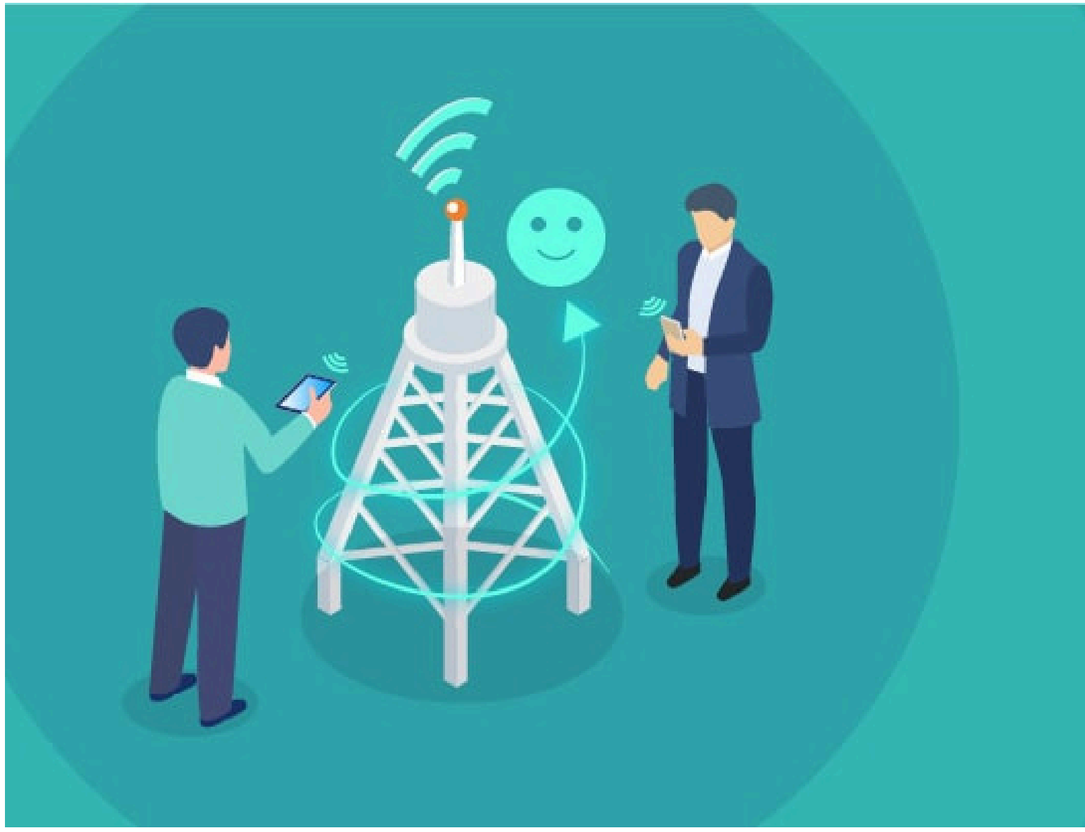
DEPI GRADUTION PROJECT

customer churn prediction

Predict behavior to retain customers



our team

01 Nada Mohamed

02 Haidy Hassan

03 yousra naser

04 Nada Kamal

05 mariam hossam

06 shourk ahmed

supervised by :

Eng. Hussein Zayed

Agenda

1. introduction
2. EDA
3. Data visualization
4. Data preprocessing
5. Machine learning
6. Deployment

Introduction

- the competitive telecom industry, retaining customers is more cost-effective than acquiring new ones. Customer churn—when customers leave a service provider for a competitor—poses a significant threat to revenue and market share. Identifying at-risk customers before they churn is crucial for implementing effective retention strategies. heading

- In this data science project, we focus on building a predictive model that can accurately identify customers who are likely to churn. Using Customer account information and Services that customer has signed up for

EDA

Exploratory Data Analysis.

01

Data collection

02

Data cleaning

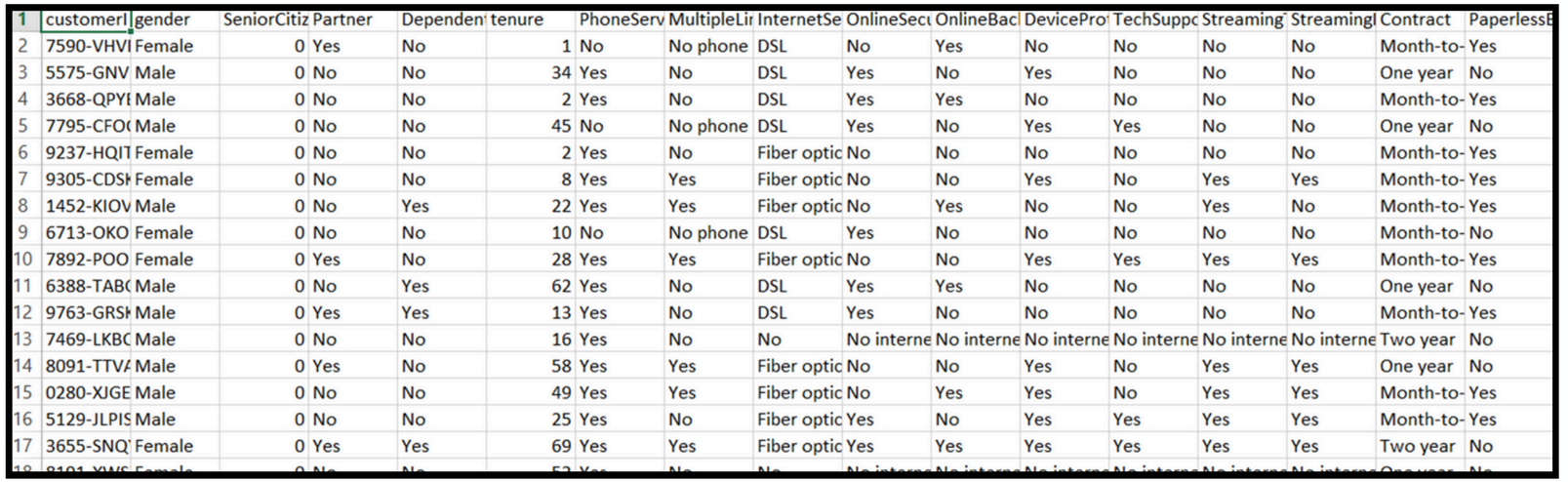
03

Feature relationships and correlation

Data collection

- Dataset Overview

This dataset contains 7,043 rows and 21 columns, detailing customer information for a telecom company



1	customerI	gender	SeniorCitiz	Partner	Dependen	tenure	PhoneServ	MultipleLir	InternetSe	OnlineSecu	OnlineBac	DevicePro	TechSuppo	Streaming	StreamingI	Contract	PaperlessB
2	7590-VHVI	Female	0	Yes	No	1	No	No phone	DSL	No	Yes	No	No	No	No	Month-to-	Yes
3	5575-GNV	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	No	No	No	One year	No
4	3668-QPYI	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	No	No	No	Month-to-	Yes
5	7795-CFO(	Male	0	No	No	45	No	No phone	DSL	Yes	No	Yes	Yes	No	No	One year	No
6	9237-HQIT	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	No	No	No	Month-to-	Yes
7	9305-CDSI	Female	0	No	No	8	Yes	Yes	Fiber optic	No	No	Yes	No	Yes	Yes	Month-to-	Yes
8	1452-KIOV	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	Yes	No	No	Yes	No	Month-to-	Yes
9	6713-OKO	Female	0	No	No	10	No	No phone	DSL	Yes	No	No	No	No	No	Month-to-	No
10	7892-POO	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	No	Yes	Yes	Yes	Yes	Month-to-	Yes
11	6388-TABC	Male	0	No	Yes	62	Yes	No	DSL	Yes	Yes	No	No	No	No	One year	No
12	9763-GRSI	Male	0	Yes	Yes	13	Yes	No	DSL	Yes	No	No	No	No	No	Month-to-	Yes
13	7469-LKBC	Male	0	No	No	16	Yes	No	No	No interne	No interne	No interne	No interne	No interne	No interne	Two year	No
14	8091-TTVA	Male	0	Yes	No	58	Yes	Yes	Fiber optic	No	No	Yes	No	Yes	Yes	One year	No
15	0280-XJGE	Male	0	No	No	49	Yes	Yes	Fiber optic	No	Yes	Yes	No	Yes	Yes	Month-to-	Yes
16	5129-JLPIS	Male	0	No	No	25	Yes	No	Fiber optic	Yes	No	Yes	Yes	Yes	Yes	Month-to-	Yes
17	3655-SNQ'	Female	0	Yes	Yes	69	Yes	Yes	Fiber optic	Yes	Yes	Yes	Yes	Yes	Yes	Two year	No

Data collection

The data set includes information about:

- Customers who left within the last month – the column is called Churn
- Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies
- Customer account information – how long they've been a customer, contract, payment method, paperless billing, monthly charges, and total charges
- Demographic info about customers – gender, age range, and if they have partners and dependents

Data collection

Read and check data set information :

```
df = pd.read_csv('/content/WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 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  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

```
df.describe()
```

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

Data cleaning

1. Handle missing values

```
df['TotalCharges'] = pd.to_numeric(df.TotalCharges, errors='coerce')
df.isnull().sum()
```

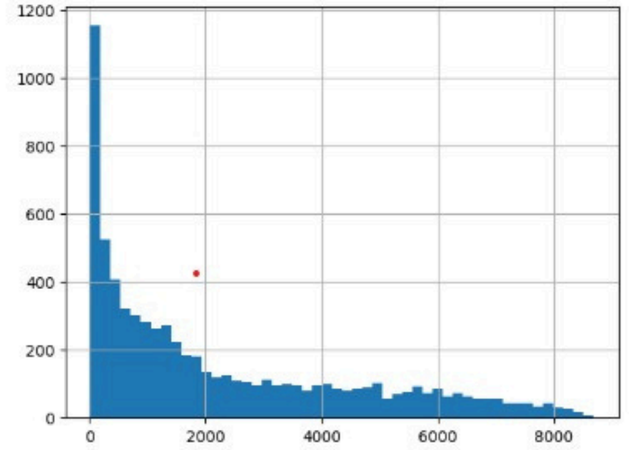
	0
gender	0
SeniorCitizen	0
Partner	0
Dependents	0
tenure	0
PhoneService	0
MultipleLines	0
InternetService	0
OnlineSecurity	0
OnlineBackup	0
DeviceProtection	0
TechSupport	0
StreamingTV	0
StreamingMovies	0
Contract	0
PaperlessBilling	0
PaymentMethod	0
MonthlyCharges	0
TotalCharges	11
Churn	0

dtype: int64

2. Fix structural errors

```
df['TotalCharges'].hist(bins=50)
```

```
<Axes: >
```




```
[14] df['TotalCharges'].mean()
```

```
2283.300440841866
```

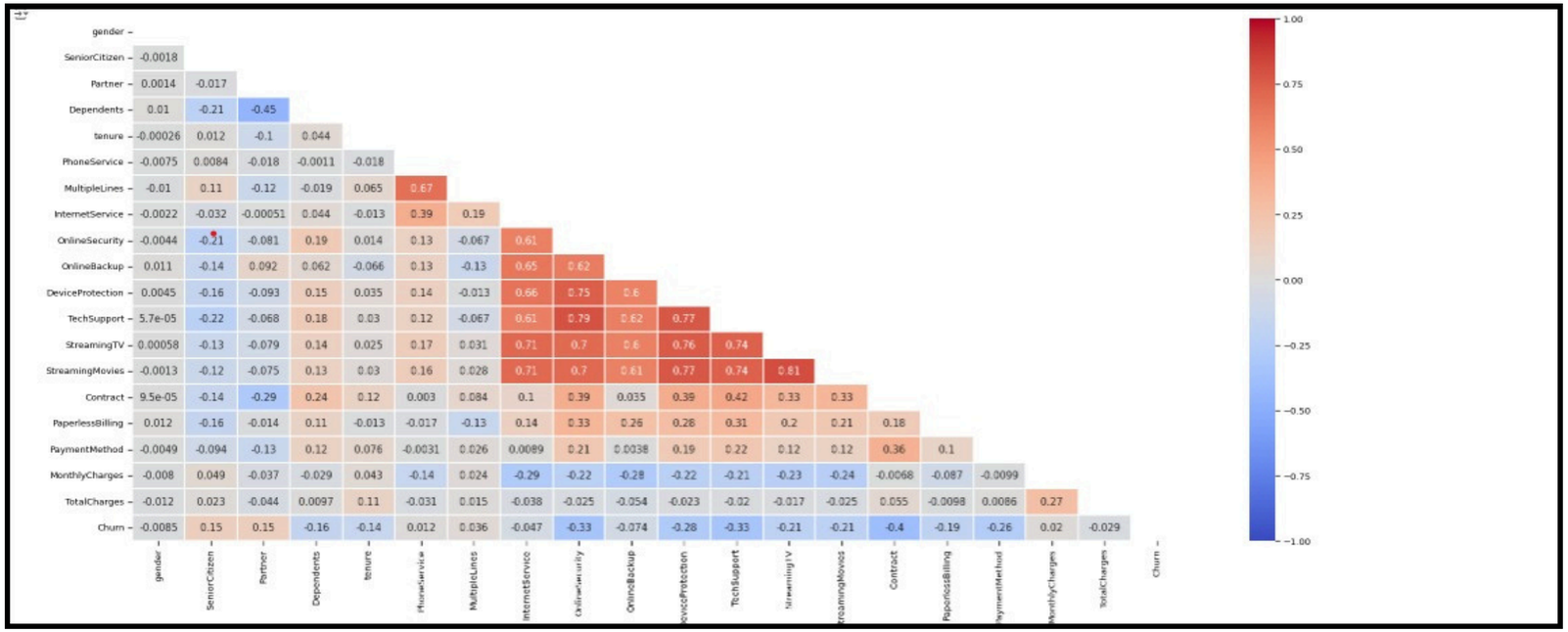
```
[15] df['TotalCharges'].median()
```

```
1397.475
```

```
[16] df.fillna(df["TotalCharges"].mean())
```

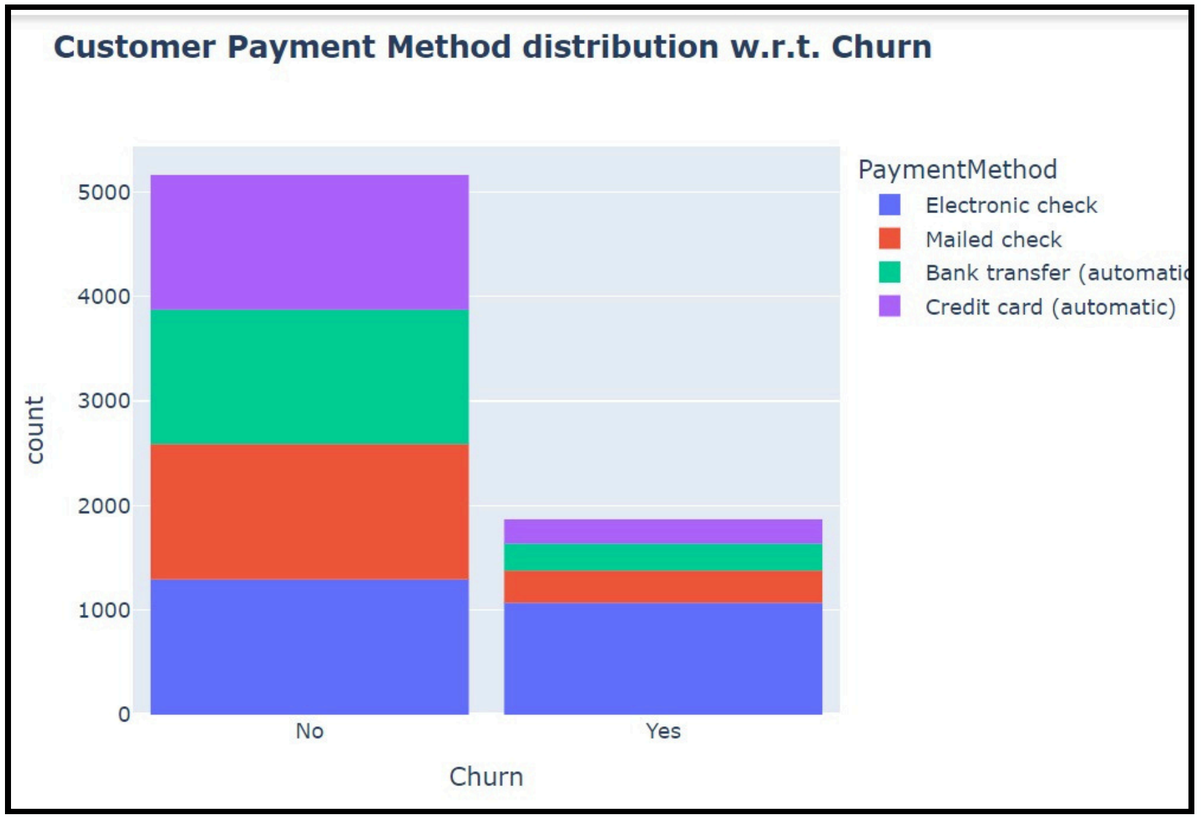
Feature correlation

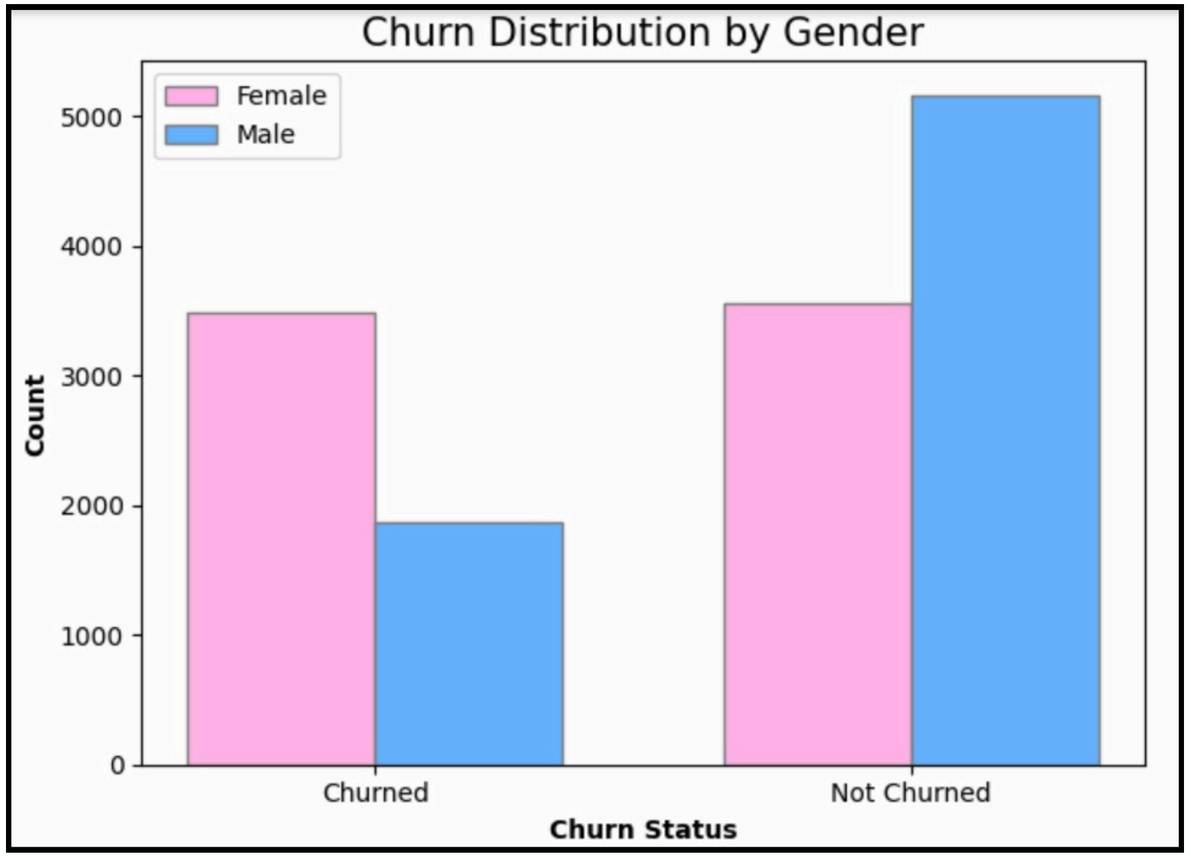
1. Correlation matrix: Calculate and visualize correlations between numerical features to understand their relationships



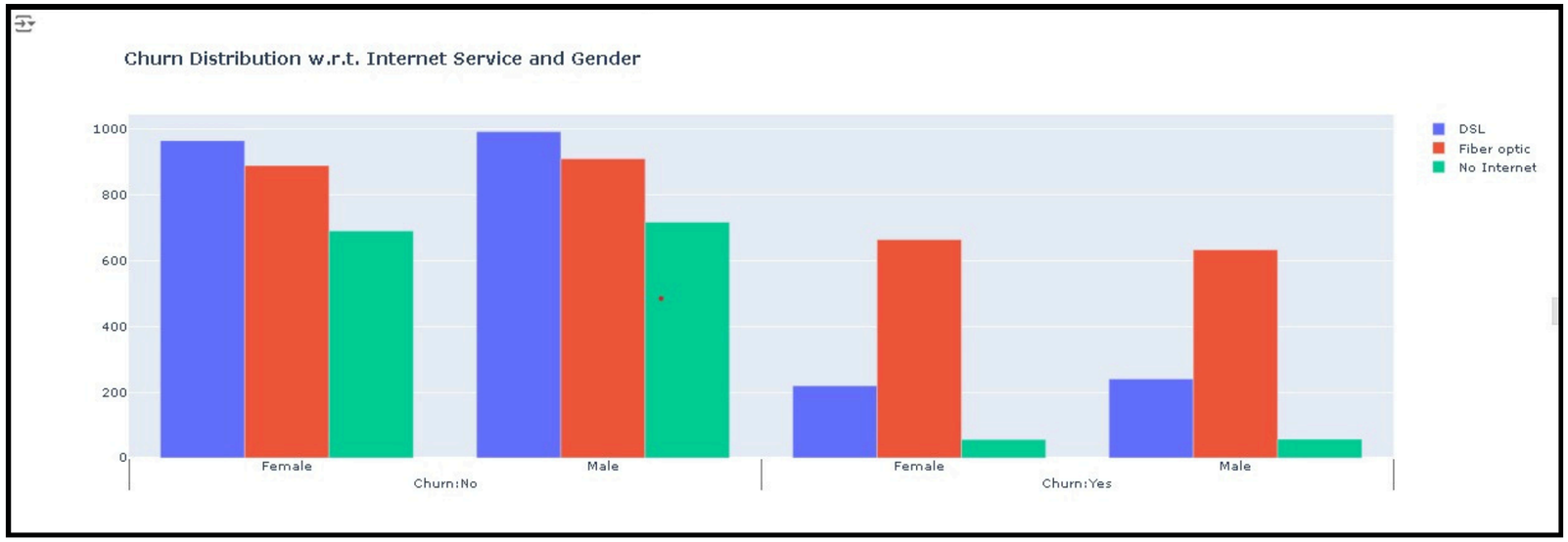
Data visualization

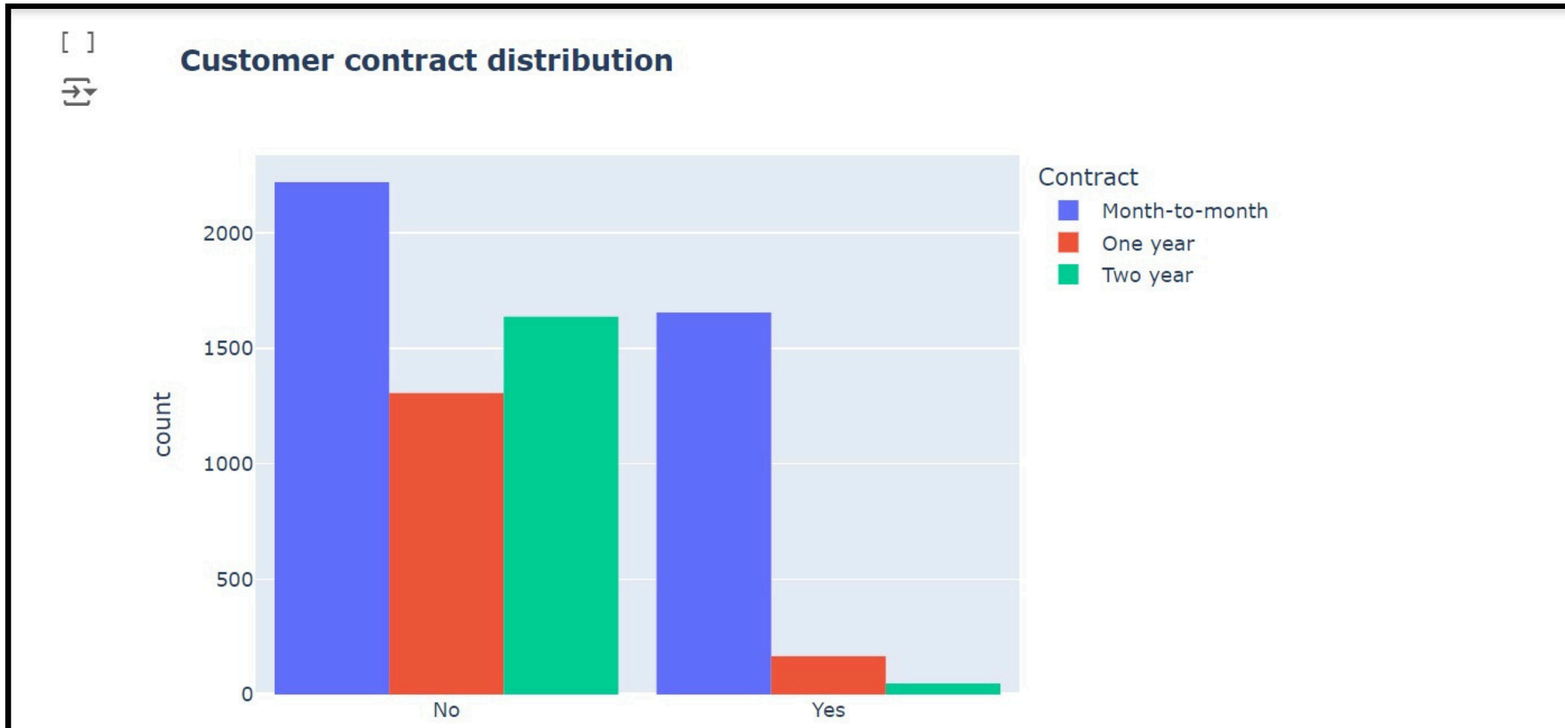
is the process of representing data graphically, allowing insights and patterns to be easily understood and communicated





Data visualization





Data preprocessing

2. Data Splitting

Split the dataset into training and test sets to evaluate the performance of the model on unseen data

```
[48] X = df.drop(columns = ['Churn'])
     y = df['Churn'].values
```

```
from sklearn.model_selection import train_test_split,GridSearchCV
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.30, random_state = 40, stratify=y)
```

3.Encoding Categorical Data

```
[42] def object_to_int(dataframe_series):
         if dataframe_series.dtype=='object':
             dataframe_series = LabelEncoder().fit_transform(dataframe_series)
         return dataframe_series
```

```
from sklearn.preprocessing import LabelEncoder
df = df.apply(lambda x: object_to_int(x))
df.head()
```

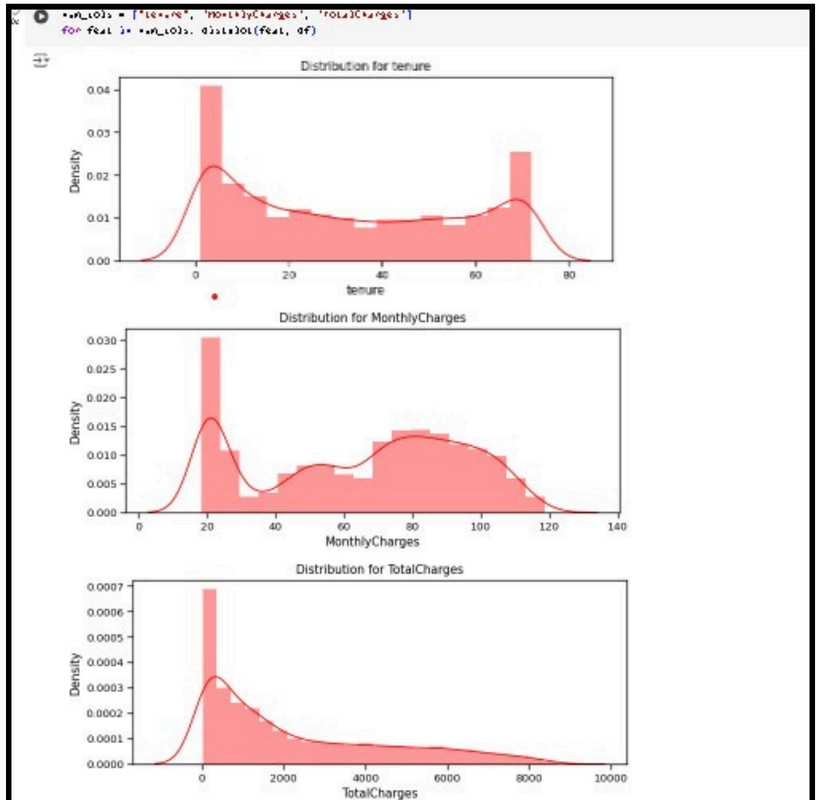
	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract
0	0	0	1	0	1	0	1	0	0	2	0	0	0	0	0
1	1	0	0	0	34	1	0	0	2	0	2	0	0	0	1
2	1	0	0	0	2	1	0	0	2	2	0	0	0	0	0
3	1	0	0	0	45	0	1	0	2	0	2	2	0	0	1
4	0	0	0	0	2	1	0	1	0	0	0	0	0	0	0

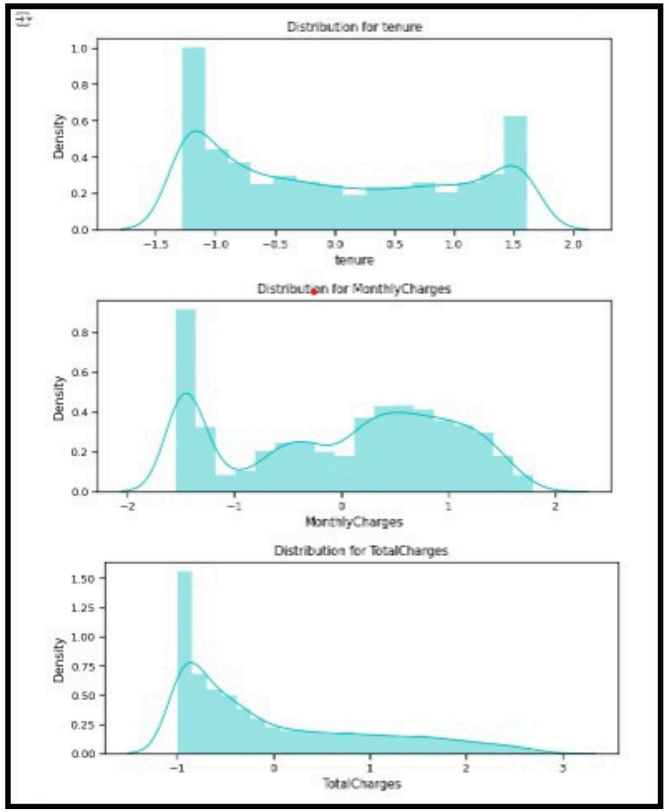
Data preprocessing

is a key step in preparing raw data for analysis or machine learning models

1. Feature Scaling

```
#Standardizing numeric attributes
from sklearn.preprocessing import StandardScaler
df_std = pd.DataFrame(StandardScaler().fit_transform(df[num_cols].astype('float64')),
                      columns=num_cols)
for feat in numerical_cols: distplot(feat, df_std, color='c')
```

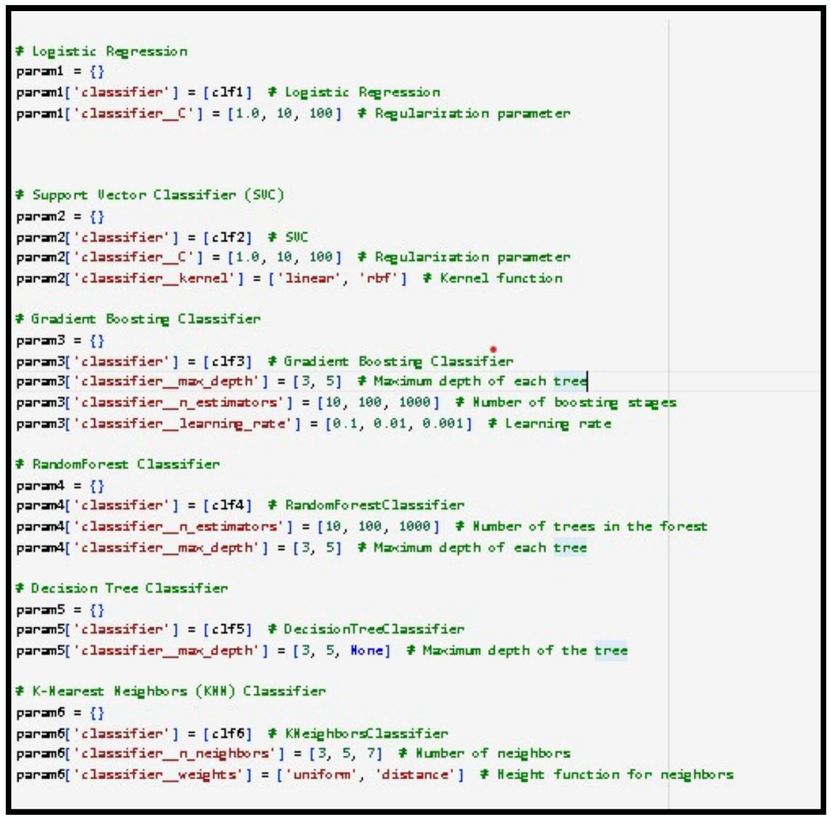




Machine learing

We use Logistic Regression, SVC, Gradient ,DecisionTree

```
# Import necessary classifiers from scikit-learn
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier

# Define classifiers
clf1 = LogisticRegression()           # Logistic Regression
clf2 = SVC()                           # Support Vector Classifier (SVC)
clf3 = GradientBoostingClassifier()    # Gradient Boosting Classifier
clf4 = RandomForestClassifier()        # Random Forest Classifier
clf5 = DecisionTreeClassifier()        # Decision Tree Classifier
clf6 = KNeighborsClassifier()          # K-Nearest Neighbors Classifier
```



```
# Logistic Regression
param1 = {}
param1['classifier'] = [clf1]  # Logistic Regression
param1['classifier__C'] = [1.0, 10, 100]  # Regularization parameter



# Support Vector Classifier (SVC)
param2 = {}
param2['classifier'] = [clf2]  # SVC
param2['classifier__C'] = [1.0, 10, 100]  # Regularization parameter
param2['classifier__kernel'] = ['linear', 'rbf']  # Kernel function

# Gradient Boosting Classifier
param3 = {}
param3['classifier'] = [clf3]  # Gradient Boosting Classifier
param3['classifier__max_depth'] = [3, 5]  # Maximum depth of each tree
param3['classifier__n_estimators'] = [10, 100, 1000]  # Number of boosting stages
param3['classifier__learning_rate'] = [0.1, 0.01, 0.001]  # Learning rate

# RandomForest Classifier
param4 = {}
param4['classifier'] = [clf4]  # RandomForestClassifier
param4['classifier__n_estimators'] = [10, 100, 1000]  # Number of trees in the forest
param4['classifier__max_depth'] = [3, 5]  # Maximum depth of each tree

# Decision Tree Classifier
param5 = {}
param5['classifier'] = [clf5]  # DecisionTreeClassifier
param5['classifier__max_depth'] = [3, 5, None]  # Maximum depth of the tree

# K-Nearest Neighbors (KNN) Classifier
param6 = {}
param6['classifier'] = [clf6]  # KNeighborsClassifier
param6['classifier__n_neighbors'] = [3, 5, 7]  # Number of neighbors
param6['classifier__weights'] = ['uniform', 'distance']  # Height function for neighbors
```

Machine learing

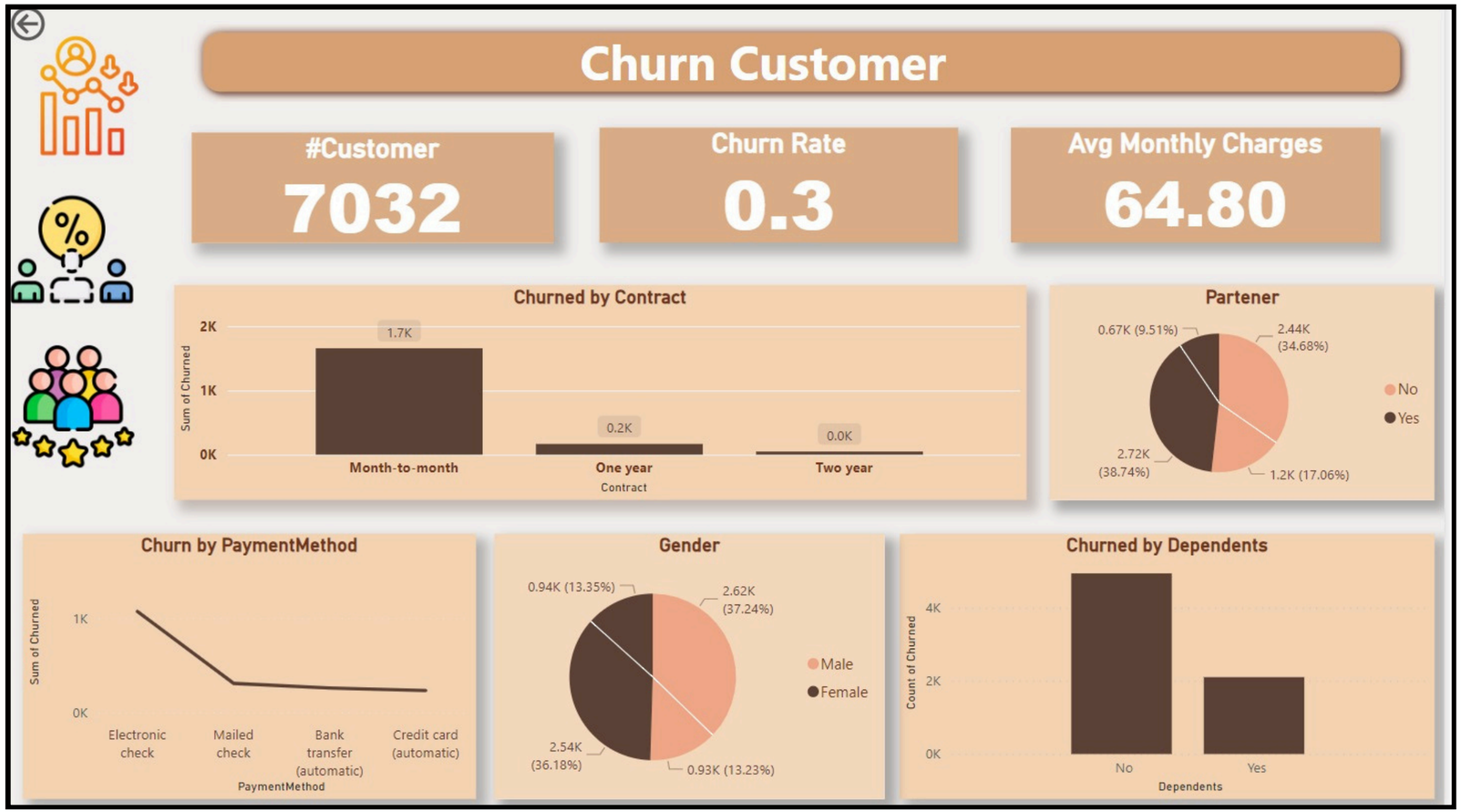
Best result in Logistic Regression

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import mean_squared_error, r2_score, f1_score,accuracy_score

# Initialize the classifier with the best parameters
best_model = GradientBoostingClassifier(
    learning_rate=0.01,
    max_depth=3,
    n_estimators=1000
)

# Fit the best model on the training data
best_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred_best = best_model.predict(X_test)

# Evaluate the model
mse_best = mean_squared_error(y_test, y_pred_best)
r2_best = r2_score(y_test, y_pred_best)
fscore_best = f1_score(y_test, y_pred_best)
accuracy_score_best = accuracy_score(y_test, y_pred_best)
print(f"Mean Squared Error with Best Model: {mse_best}")
print(f"R^2 Score with Best Model: {r2_best}")
print (f"R^2 Score with Best Model: {fscore_best}")
print (f"Accuracy Score with Best Model: {accuracy_score_best}")
```

```
Mean Squared Error with Best Model: 0.1895734597156398
R^2 Score with Best Model: 0.028756405432059595
R^2 Score with Best Model: 0.6078431372549019
Accuracy Score with Best Model: 0.8104265402843602
```

Dash board

using power BI



Deployment

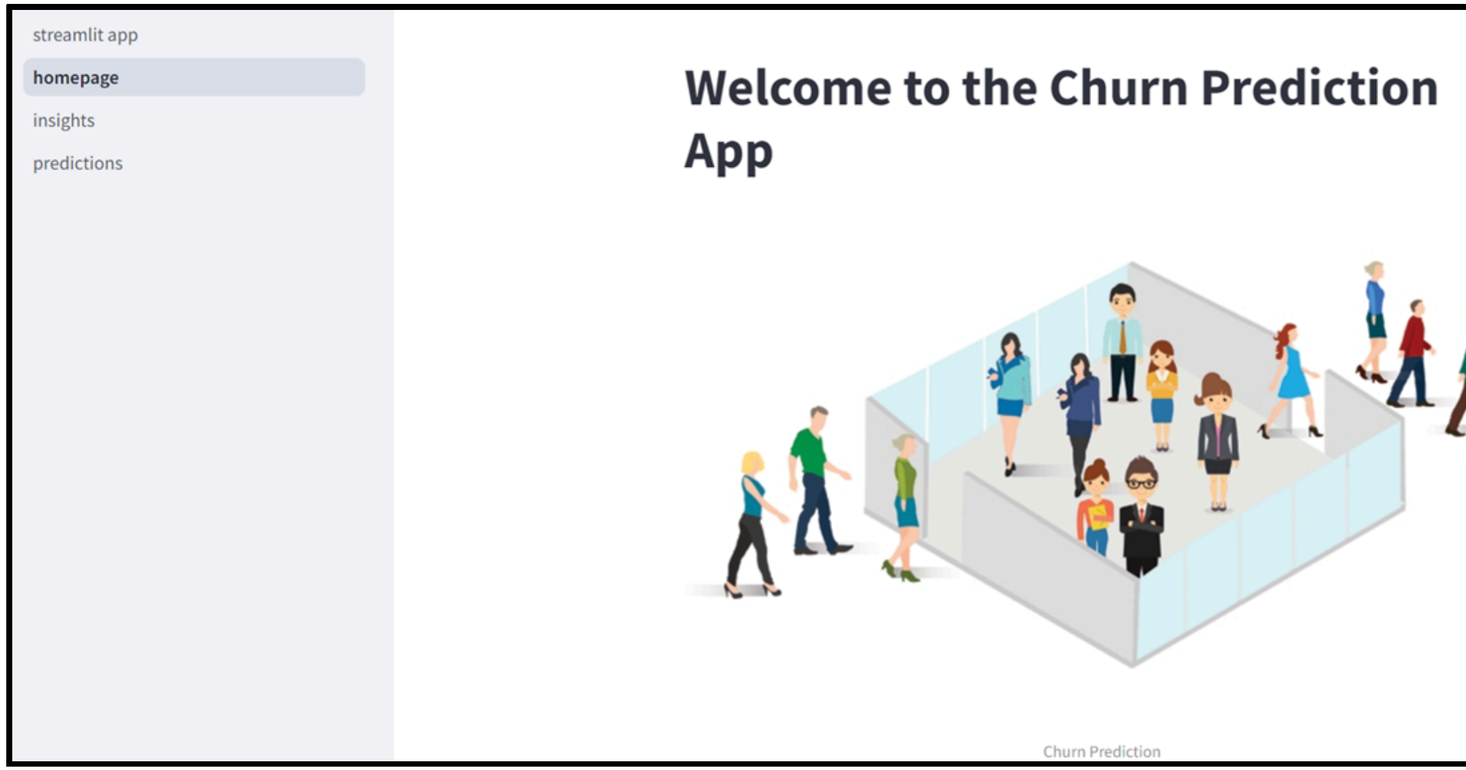
using streamlit

Home page content:

1.Why is this important?

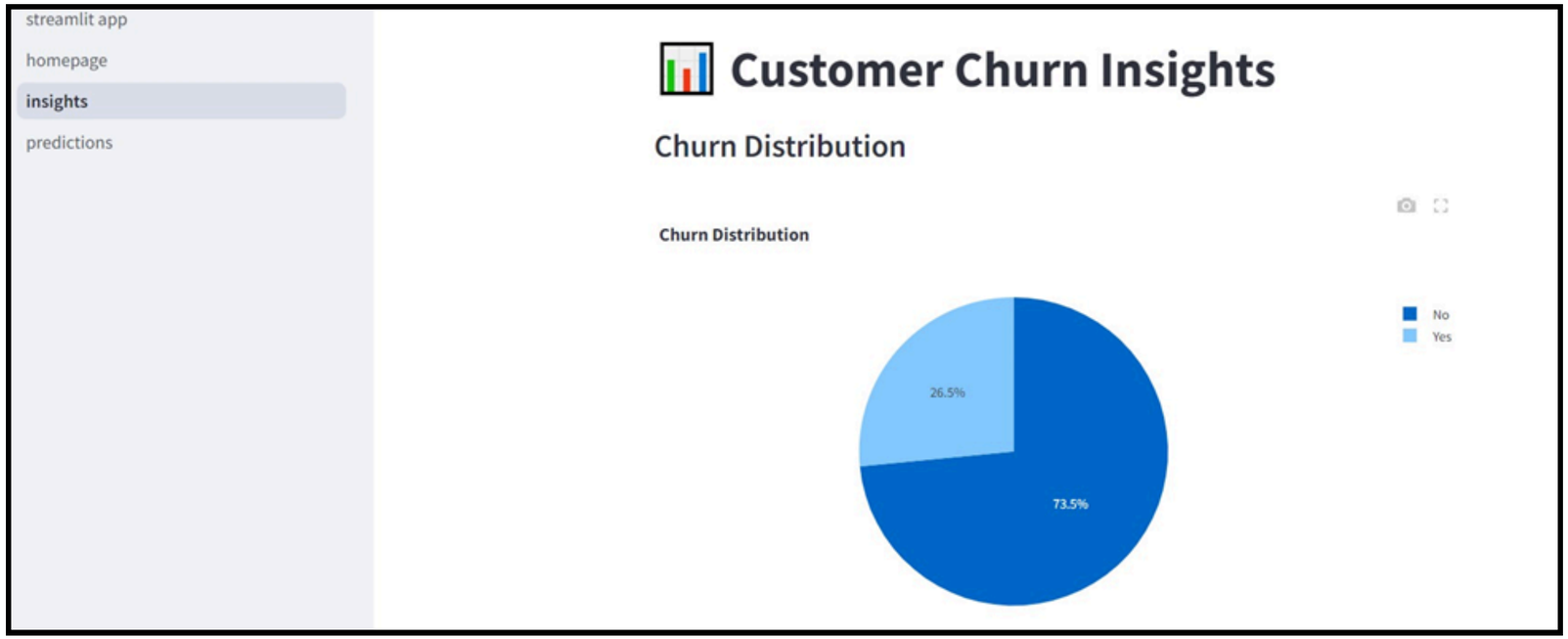
2.Why is this important?

3.What will this app take from you?



Deployment

Customer Churn Insights



Deployment

Churn Prediction

THANK YOU