Customer Churn Prediction Project Documentation

Project Objective

The objective of this project is to develop a customer churn prediction model that helps businesses identify customers who are likely to churn, enabling them to take proactive measures to retain those customers. The project aims to reduce customer churn and its associated negative impacts on a business, such as loss of revenue and market share.

Design Thinking Process

Phase 1: Problem Definition

- Defined the problem of customer churn and its significance in the business context.
- Set clear project goals and objectives.

Phase 2: Data Collection

- Gathered relevant customer data, including demographics, contract information, and usage patterns.
- Ensured data quality by addressing missing values and errors.

Phase 3: Data Preprocessing

- Pre-processed the data by handling categorical variables through one-hot encoding and scaling numerical features.
- Checked for duplicate records and ensured the dataset was clean and ready for analysis.

Phase 4: Data Analysis and Visualization

- Utilized IBM Cognos to create interactive dashboards and reports.
- Visualized churn patterns, retention rates, and key factors influencing churn.
- Presented insights on customer behaviour and identified areas for improvement.

Phase 5: Predictive Modelling

- Built predictive models using Python, specifically Random Forest and Logistic Regression.
- Evaluated model performance using metrics like accuracy, precision, recall, and F1-score.
- Provided guidance on model selection and usage.

Analysis Objectives

The analysis of this project includes:

- Visualizing churn patterns and customer demographics.

- Analyzing the impact of different contract terms on churn rates.
- Identifying key factors influencing churn.
- Developing predictive models to forecast customer churn.

Data Collection

The dataset used in this project contains customer information, including gender, contract details, usage patterns, and the churn label. It is collected from the Telco industry and represents historical customer data.

Data Visualization using IBM Cognos

IBM Cognos is employed to create interactive dashboards and reports, offering visual representations of the data. The visualizations provide insights into customer behavior, enabling businesses to make data-driven decisions to reduce churn.

Predictive Modeling

Two machine learning models, Random Forest and Logistic Regression, are developed and evaluated for their ability to predict customer churn. These models use preprocessed data and provide a basis for identifying potential churners based on historical data.

Business Impact

The insights gained from the analysis and the predictive model can help businesses reduce customer churn in several ways:

- Early identification of potential churners allows businesses to implement retention strategies, such as tailored offers or improved customer service.
- Analysis of customer demographics and behavior can guide marketing and communication efforts.
- Improved customer retention leads to increased customer lifetime value and enhanced business profitability.

Submission Details

GitHub Repository

The project's code and files are available in the GitHub repository: https://github.com/Miru-nalini/Customer-Churn-Prediction

Replication Instructions

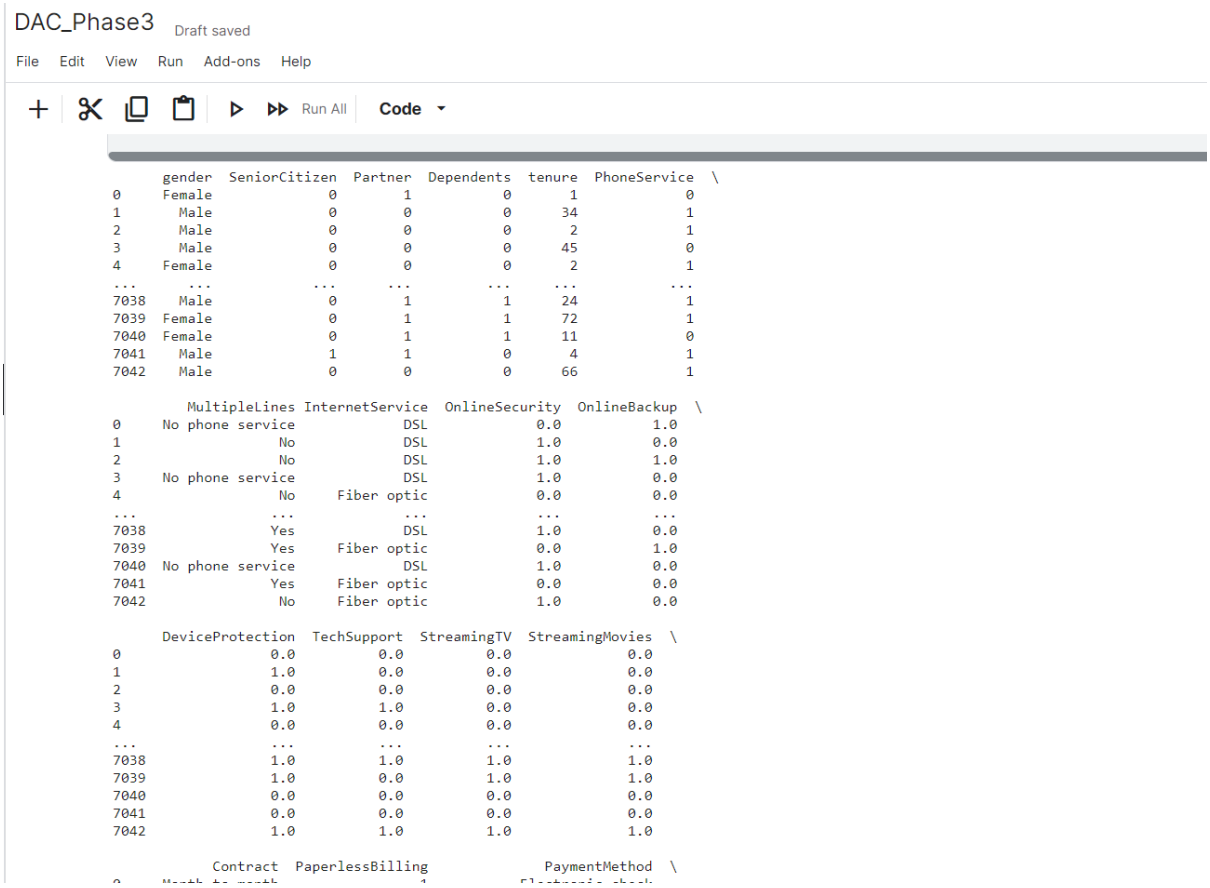
To replicate the analysis and generate visualizations using IBM Cognos and build the predictive model using Python, follow these steps:

1. Clone the GitHub repository to your local machine.
2. Install the required Python libraries using **pip install -r requirements.txt**.
3. Download the pre-processed dataset used in the project.

4. Use IBM Cognos to create interactive dashboards and reports for visualizing customer churn patterns.
5. Execute the Python script to build and evaluate the predictive models.
6. Use the provided data and example outputs to gain insights and make data-driven decisions.

Example Outputs

The documentation includes example outputs of the visualizations and model evaluation metrics, which are available in the project repository.



DAC_Phase3 Draft saved

File Edit View Run Add-ons Help

Run All Code

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	\
0	Female	0	1	0	1	0	
1	Male	0	0	0	34	1	
2	Male	0	0	0	2	1	
3	Male	0	0	0	45	0	
4	Female	0	0	0	2	1	
...	...	...	...	...	...	...	
7038	Male	0	1	1	24	1	
7039	Female	0	1	1	72	1	
7040	Female	0	1	1	11	0	
7041	Male	1	1	0	4	1	
7042	Male	0	0	0	66	1	

	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	\
0	No phone service	DSL	0.0	1.0	
1	No	DSL	1.0	0.0	
2	No	DSL	1.0	1.0	
3	No phone service	DSL	1.0	0.0	
4	No	Fiber optic	0.0	0.0	
...	...	...	...	...	
7038	Yes	DSL	1.0	0.0	
7039	Yes	Fiber optic	0.0	1.0	
7040	No phone service	DSL	1.0	0.0	
7041	Yes	Fiber optic	0.0	0.0	
7042	No	Fiber optic	1.0	0.0	

	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	\
0	0.0	0.0	0.0	0.0	
1	1.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	1.0	1.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	
...	...	...	...	...	
7038	1.0	1.0	1.0	1.0	
7039	1.0	0.0	1.0	1.0	
7040	0.0	0.0	0.0	0.0	
7041	0.0	0.0	0.0	0.0	
7042	1.0	1.0	1.0	1.0	

	Contract	PaperlessBilling	PaymentMethod	\
0	Month-to-month	1	Electronic check	

DAC_Phase3 Draft saved

File Edit View Run Add-ons Help

Run All Code

```
2                0.0          0.0          0.0              0.0
3                1.0          1.0          0.0              0.0
4                0.0          0.0          0.0              0.0
...              ...          ...          ...              ...
7038             1.0          1.0          1.0              1.0
7039             1.0          0.0          1.0              1.0
7040             0.0          0.0          0.0              0.0
7041             0.0          0.0          0.0              0.0
7042             1.0          1.0          1.0              1.0

            Contract  PaperlessBilling              PaymentMethod  \
0     Month-to-month                 1           Electronic check
1           One year                 0               Mailed check
2     Month-to-month                 1               Mailed check
3           One year                 0  Bank transfer (automatic)
4     Month-to-month                 1           Electronic check
...              ...               ...                        ...
7038        One year                 1               Mailed check
7039        One year                 1   Credit card (automatic)
7040  Month-to-month                 1           Electronic check
7041  Month-to-month                 1               Mailed check
7042        Two year                 1  Bank transfer (automatic)

      MonthlyCharges  TotalCharges  Churn
0              29.85         29.85      0
1              56.95       1889.50      0
2              53.85        108.15      1
3              42.30       1840.75      0
4              70.70        151.65      1
...              ...           ...    ...
7038           84.80       1990.50      0
7039          103.20       7362.90      0
7040           29.60        346.45      0
7041           74.40        306.60      1
7042          105.65       6844.50      0

[7043 rows x 20 columns]
```

+ Code + Markdown

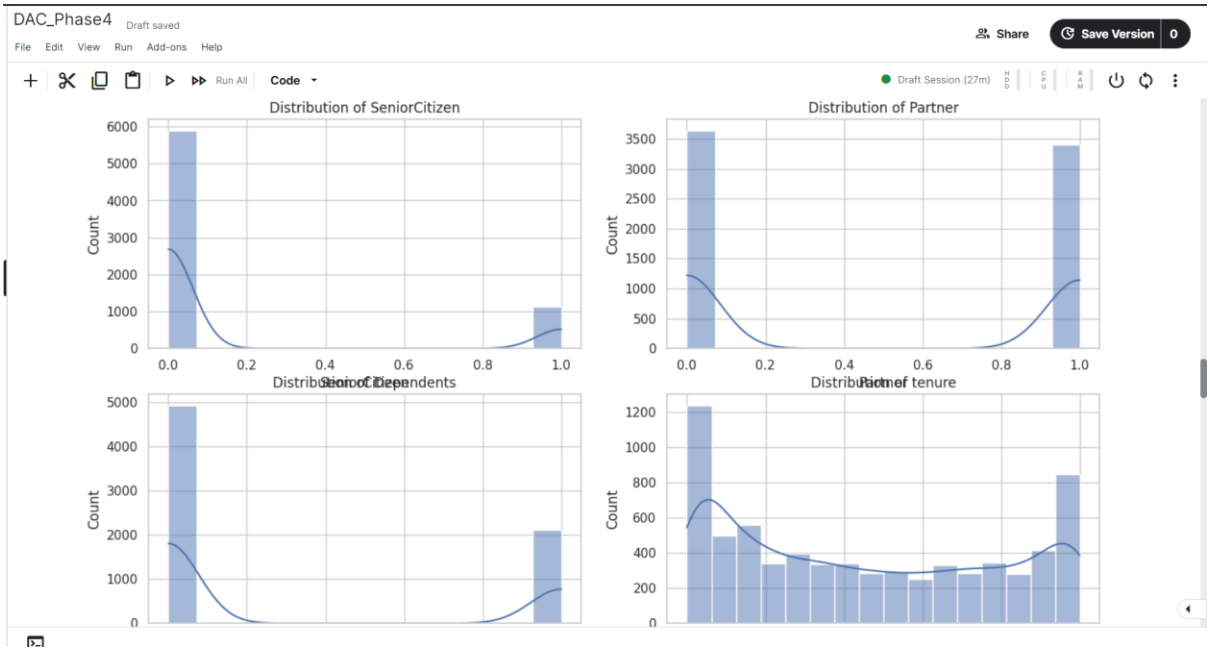
DAC_Phase3 Draft saved

File Edit View Run Add-ons Help

Run All Code

```python
from sklearn.preprocessing import MinMaxScaler

# Create a MinMaxScaler instance
scaler = MinMaxScaler()

# Define the columns you want to scale (numeric columns)
columns_to_scale = ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']

# Fit and transform the selected columns
df[columns_to_scale] = scaler.fit_transform(df[columns_to_scale])

# Display the resulting dataset with scaled values
print(df.head())
```

```
   gender  SeniorCitizen  Partner  Dependents    tenure  PhoneService  \
0  Female            0.0        1           0  0.013889             0
1    Male            0.0        0           0  0.472222             1
2    Male            0.0        0           0  0.027778             1
3    Male            0.0        0           0  0.625000             0
4  Female            0.0        0           0  0.027778             1

      MultipleLines InternetService  OnlineSecurity  OnlineBackup  \
0  No phone service             DSL             0.0           1.0
1                No             DSL             1.0           0.0
2                No             DSL             1.0           1.0
3  No phone service             DSL             1.0           0.0
4                No     Fiber optic             0.0           0.0

   DeviceProtection  TechSupport  StreamingTV  StreamingMovies  \
0               0.0          0.0          0.0              0.0
1               1.0          0.0          0.0              0.0
2               0.0          0.0          0.0              0.0
3               1.0          1.0          0.0              0.0
4               0.0          0.0          0.0              0.0
```



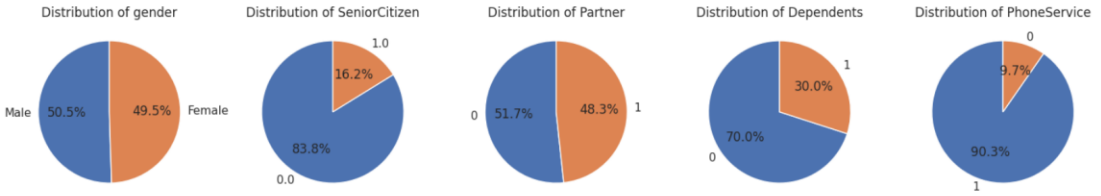
DAC_Phase4 Draft saved

File Edit View Run Add-ons Help

Share Save Version 0

Run All Code Draft Session (27m)

```
# Create pie charts for each categorical variable
for i, variable in enumerate(categorical_variables):
    ax = axes[i]
    counts = df[variable].value_counts()
    labels = counts.index
    sizes = counts.values
    ax.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
    ax.set_title(f'Distribution of {variable}')

plt.tight_layout()
plt.show()
```



Pie charts showing the distribution of each selected categorical variable.

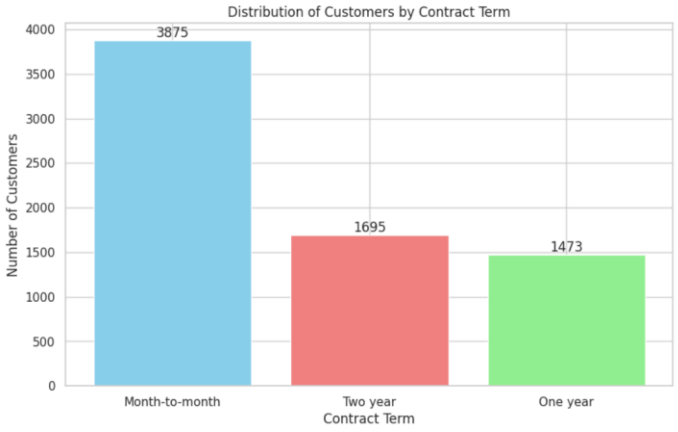
DAC_Phase4 Draft saved

File Edit View Run Add-ons Help

Share Save Version 0

Run All Code Draft Session (27m)

```
plt.show()
```

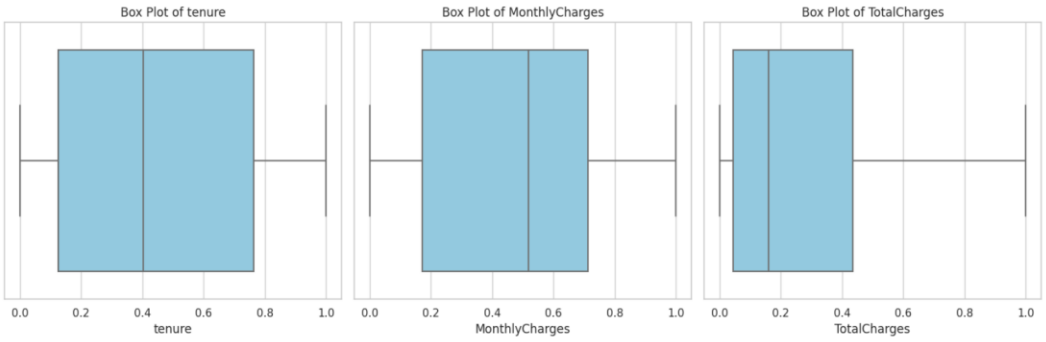


DAC_Phase4 Draft saved

File Edit View Run Add-ons Help

Share Save Version 0

Run All Code Draft Session (28m)



Box plots showing the distribution of each selected numerical variable in the dataset. Box plots provide information about the median, quartiles, and potential outliers for each variable.

[26]:

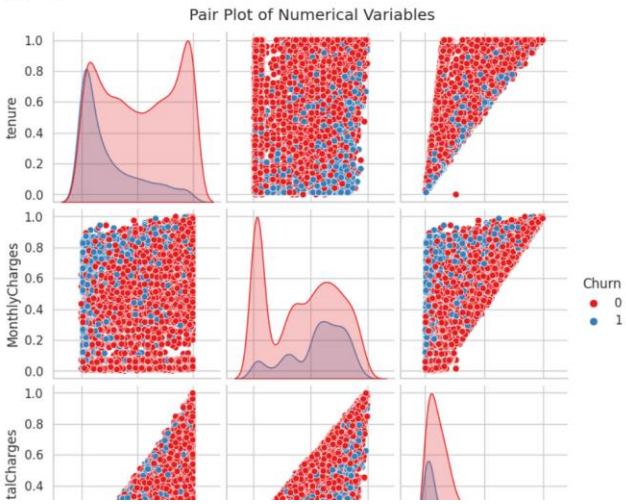
```
# Define a list of numerical variables you want to include in the pair plot
numerical_variables = ['tenure', 'MonthlyCharges', 'TotalCharges']
```

DAC_Phase4 Draft saved

File Edit View Run Add-ons Help

Share Save Version 0

Run All Code Draft Session (28m)



DAC_Phase4 Draft saved

File Edit View Run Add-ons Help

Share Save Version 0

Run All Markdown Draft Session (28m)

may be related to customer churn.

BUILDING THE MODEL

Random Forests and Logistic Regression

+ Code + Markdown

[31]:

```
df.head()
```

[31]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilli
0	Female	0.0	1	0	0.013889	0	No phone service	DSL	0.0	1.0	0.0	0.0	0.0	0.0	Month-to-month	
1	Male	0.0	0	0	0.472222	1	No	DSL	1.0	0.0	1.0	0.0	0.0	0.0	One year	
2	Male	0.0	0	0	0.027778	1	No	DSL	1.0	1.0	0.0	0.0	0.0	0.0	Month-to-month	
3	Male	0.0	0	0	0.625000	0	No phone service	DSL	1.0	0.0	1.0	1.0	0.0	0.0	One year	
4	Female	0.0	0	0	0.027778	1	No	Fiber optic	0.0	0.0	0.0	0.0	0.0	0.0	Month-to-month	

+ Code + Markdown

[32]:

```
import numpy as np
import pandas as pd
```

DAC_Phase4 Draft saved

File Edit View Run Add-ons Help

Run All Markdown

```
print("\nLogistic Regression Model Results:")
print("Accuracy:", accuracy_score(y_test, lr_predictions))
print("Classification Report:")
print(classification_report(y_test, lr_predictions))
print("Confusion Matrix:")
print(confusion_matrix(y_test, lr_predictions))
```

```
Random Forest Model Results:
Accuracy: 0.7970191625266146
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1036
           1       0.66      0.48      0.55       373

    accuracy                           0.80      1409
   macro avg       0.75      0.69      0.71      1409
weighted avg       0.78      0.80      0.79      1409

Confusion Matrix:
[[945  91]
 [195 178]]

Logistic Regression Model Results:
Accuracy: 0.8204400283889283
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.90      0.88      1036
           1       0.69      0.59      0.64       373

    accuracy                           0.82      1409
   macro avg       0.77      0.75      0.76      1409
weighted avg       0.81      0.82      0.82      1409

Confusion Matrix:
[[935 101]
 [152 221]]
```