INTRODUCTION TO DATA SCIENCE

TOPIC:- EXPLORATORY DATA ANALYSIS

BY : AMAN SINGH

1. Abstract
2. Introduction
3. Dataset
4. Exploratory Data Analysis/Preprocessing
5. Feature Engineering
6. Visualization
7. Results
8. Conclusion
9. References
10. GitHub Repository for Code(optional)

ABSTRACT

- Exploratory Data Analysis (EDA) is a fundamental step in the data science workflow that involves the initial investigation of data to discover patterns, detect anomalies, test hypotheses, and check assumptions through summary statistics and graphical representations. By employing various statistical tools and visualization techniques, EDA enables analysts to understand the underlying structure of the data, identify significant variables, and uncover relationships between them. This process is crucial for guiding subsequent data processing, feature engineering, and model selection in machine learning projects. EDA not only enhances the quality and efficiency of data-driven decision-making but also ensures that the insights derived are robust and reliable. Through systematic exploration, EDA lays the groundwork for building predictive models that accurately reflect the complexities of real-world data.

INTRODUCTION

Exploratory Data Analysis is an approach to analyze the datasets to summarize their main characteristics in form of visual methods.

EDA is nothing but a data exploration technique to understand various aspects of the data.

The main aim of EDA is to obtain confidence in a data to an extent where we are ready to engage a machine learning model.

EDA is important to analyze the data it's a first steps in data analysis process.

EDA give a basic idea to understand the data and make sense of the data to figure out the question you need to ask and find out the best way to manipulate the dataset to get the answer to your question.

Exploratory data analysis help us to finding the errors, discovering data, mapping out data structure, finding out anomalies.

Exploratory data analysis is important for business process because we are preparing dataset for deep thorough analysis that will detect you business problem.

EDA help to build a quick and dirty model, or a baseline model, which can serve as a comparison against later models that you will build.

```
# File display the data types of all the columns
telco_base_data.dtypes
```

```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

DATASET

HEAD OF THE DATASET

```
[7] telco_base_data.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Int
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	
4	9237-HQITU	Female	0	No	No	2	Yes	No	

5 rows × 21 columns

TAIL OF THE DATASET

```
telco_base_data.tail()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes
7042	3186-AJIEK	Male	0	No	No	66	Yes	No

5 rows × 21 columns

Add code cell
⌘/Ctrl+M B

DATA SHAPE AND COLUMNS VALUES

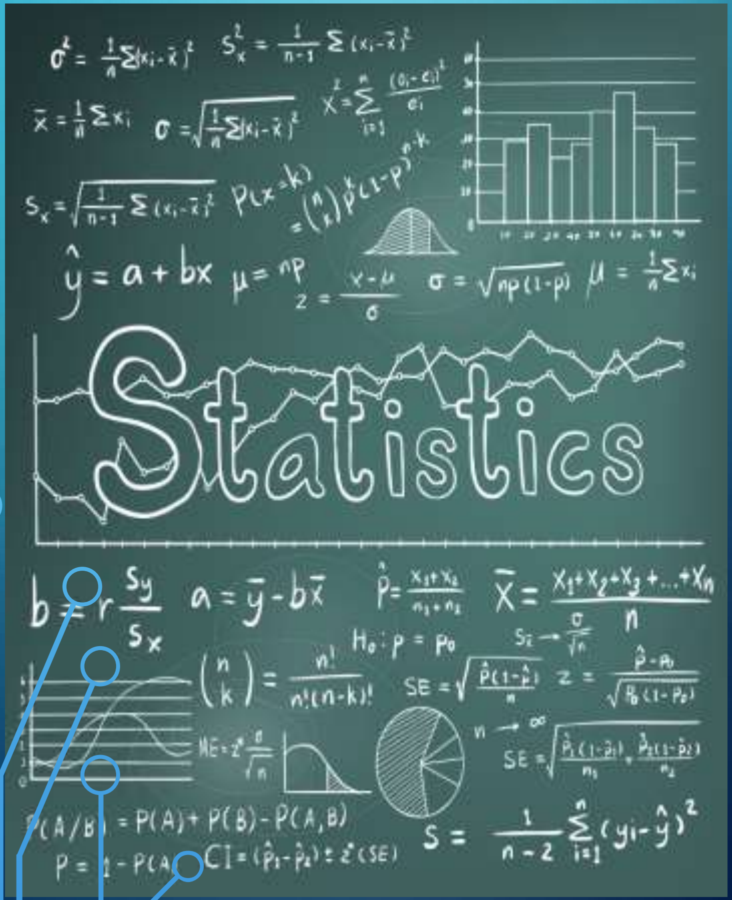
```
[8] telco_base_data.shape

(7043, 21)

[9] telco_base_data.columns.values

array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

STATISTICS OF NUMERIC VARIABLES



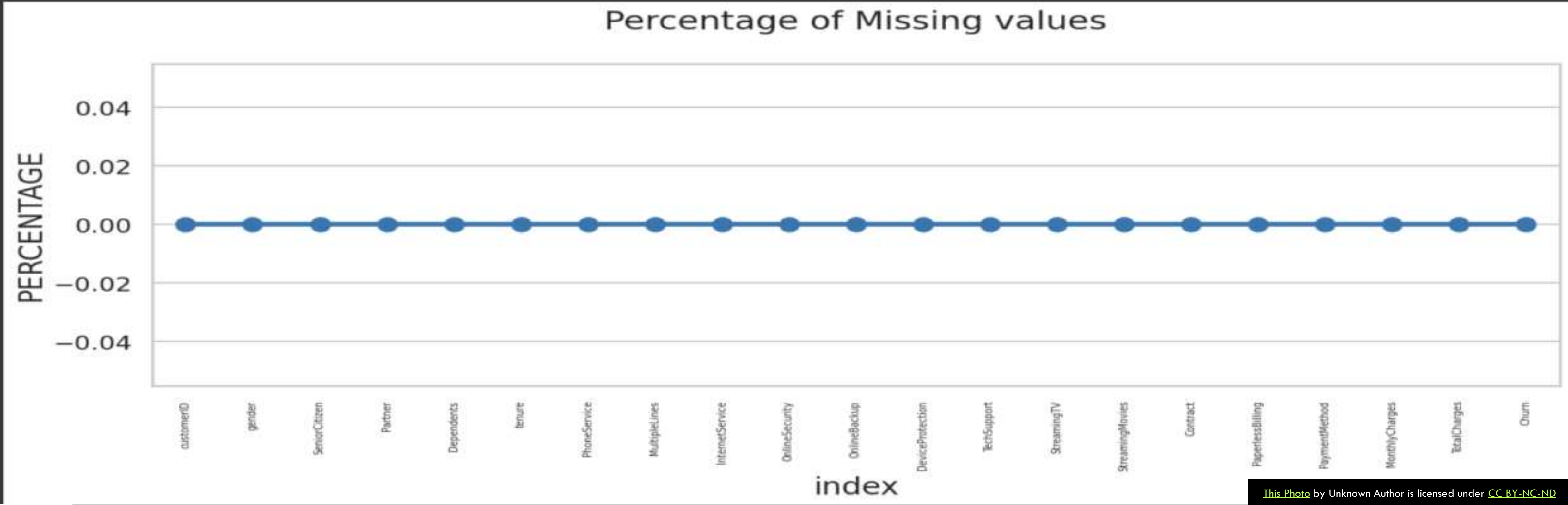

```
# Check the descriptive statistics of numeric variables
telco_base[illegible]cribe()
```

File display

	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

CHECKING MISSING VALUES

```
missing = pd.DataFrame((telco_base_data.isnull().sum())*100/telco_base_data.shape[0]).reset_index()
plt.figure(figsize=(16,5))
# Use column names from the 'missing' DataFrame for x and y arguments
ax = sns.pointplot(x='index', y=0, data=missing)
plt.xticks(rotation =90,fontsize =7)
plt.title("Percentage of Missing values")
plt.ylabel("PERCENTAGE")
plt.show()
```



CHECKING NULL VALUES

```
telco_data.isnull().sum()
```

	0
customerID	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

```
telco_data.loc[telco_data ['TotalCharges'].isnull() == True]
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSuppo
488	4472-LVYGI	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	Yes	Y
753	3115-CZMZD	Male	0	No	Yes	0	Yes	No	No	No internet service	...	No internet service	No intern servi
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	...	Yes	
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No intern servi
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	Yes	Y
3331	7644-OMVMY	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No intern servi
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No intern servi
4380	2520-SGTTA	Female	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No intern servi
5218	2923-ARZLG	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No intern servi
6670	4075-WKNIU	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	...	Yes	Y
6754	2775-SEFEE	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	...	No	Y

11 rows × 21 columns

4. Missing Value Treatement

Since the % of these records compared to total dataset is very low ie 0.15%, it is safe to ignore them from further processing.

```
#Removing missing values
telco_data.dropna(how = 'any', inplace = True)

#telco_data.fillna(0)
```

+ Code + Text

5. Divide customers into bins based on tenure e.g. for tenure < 12 months: assign a tenure group if 1-12, for tenure between 1 to 2 Yrs, tenure group of 13-24; so on...

```
[23] # Get the max tenure
     print(telco_data['tenure'].max())
```

72

```
# Group the tenure in bins of 12 months
labels = ["{0} - {1}".format(i, i + 11) for i in range(1, 72, 12)]

telco_data['tenure_group'] = pd.cut(telco_data.tenure, range(1, 80, 12), right=False, labels=labels)
```

```
[25] telco_data['tenure_group'].value_counts()
```

tenure_group	count
1 - 12	2175
61 - 72	1407
13 - 24	1024
25 - 36	832
49 - 60	832
37 - 48	762

dtype: int64

6. Remove columns not required for processing

```
#drop column customerID and tenure
telco_data.drop(columns= ['customerID','tenure'], axis=1, inplace=True)
telco_data.head()
```

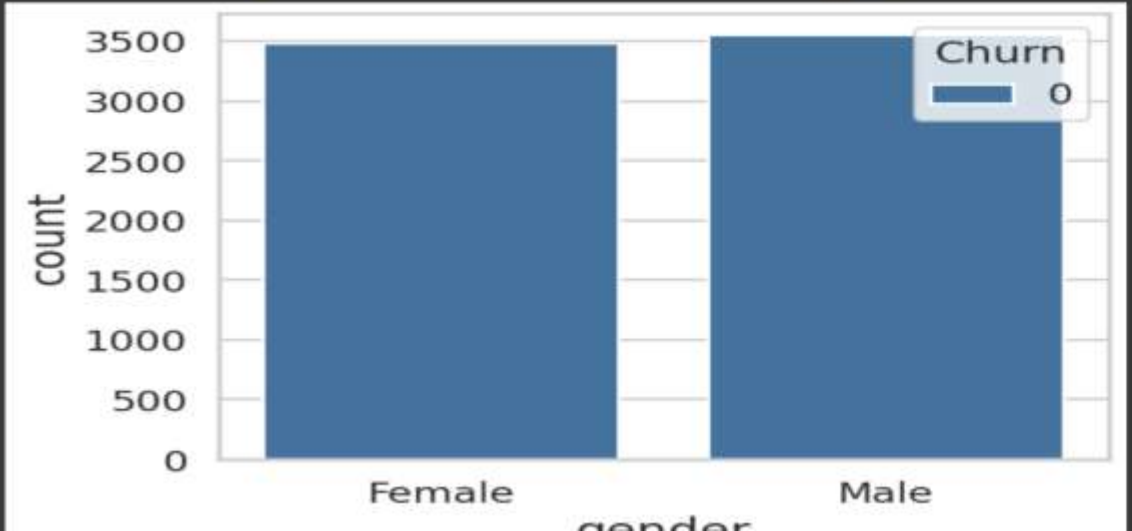
	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Streaming
0	Female	0	Yes	No	No	No phone service	DSL	No	Yes	No	No	N
1	Male	0	No	No	Yes	No	DSL	Yes	No	Yes	No	N
2	Male	0	No	No	Yes	No	DSL	Yes	Yes	No	No	N
3	Male	0	No	No	No	No phone service	DSL	Yes	No	Yes	Yes	N
4	Female	0	No	No	Yes	No	Fiber optic	No	No	No	No	N

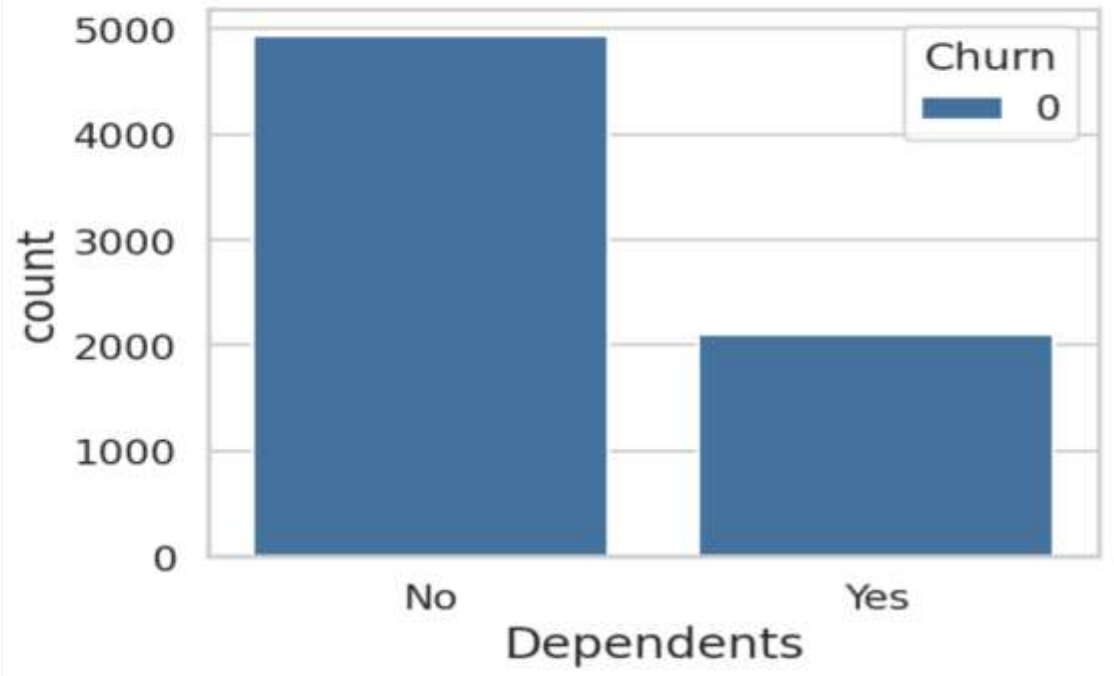
Data Exploration

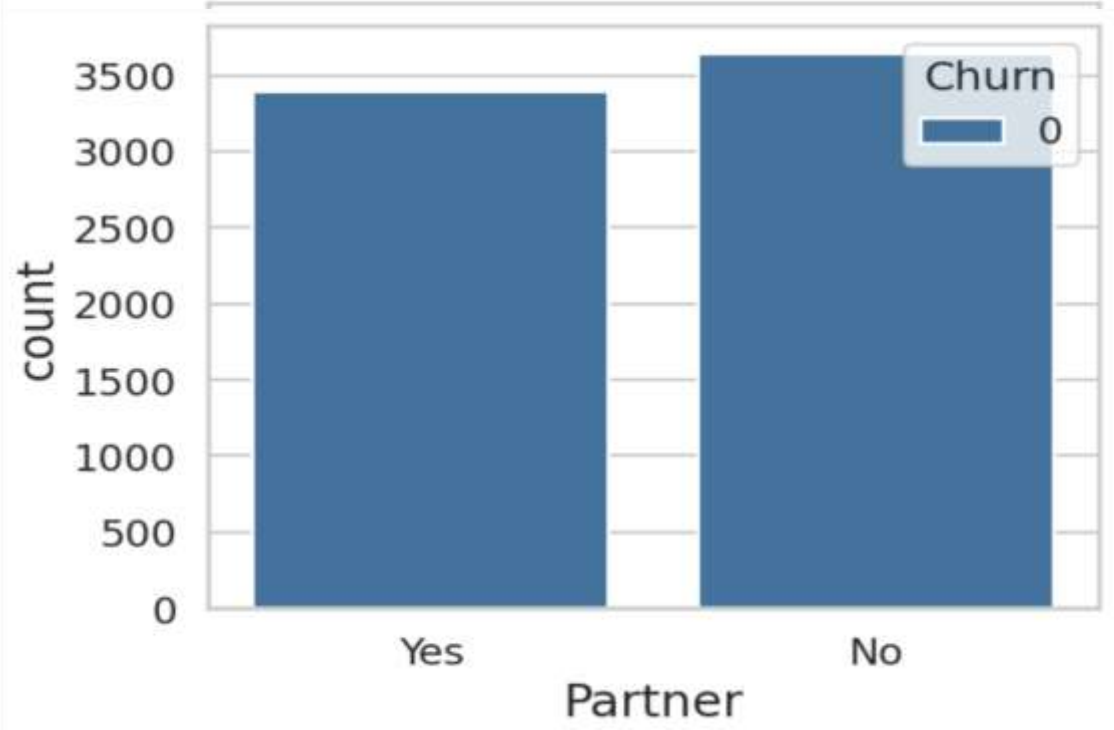
*7. *Plot distibution of individual predictors by churn

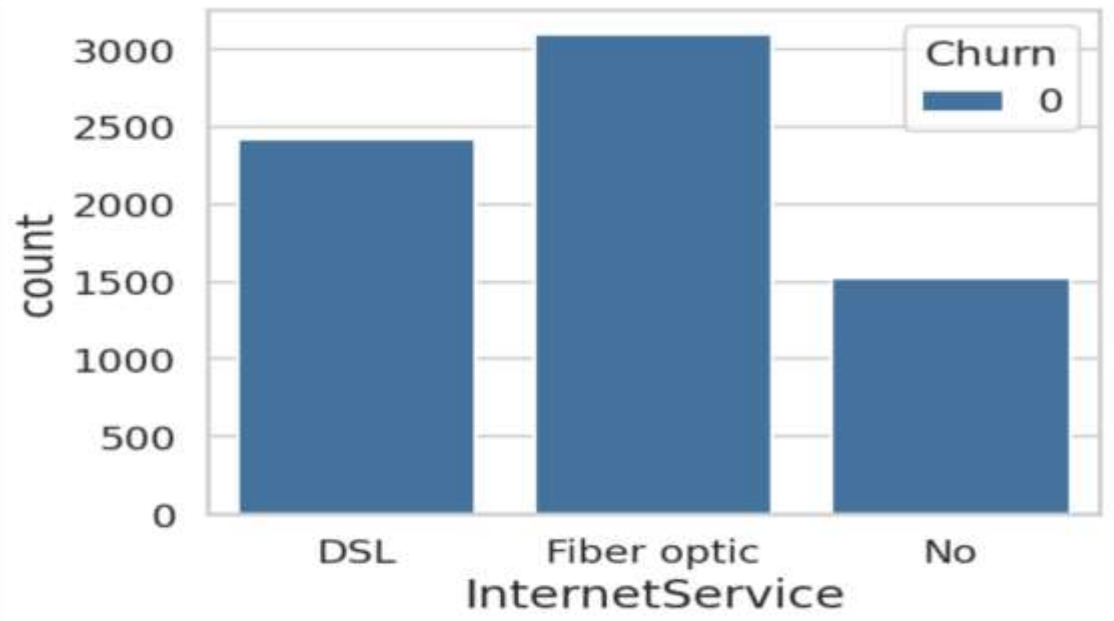
Univariate Analysis

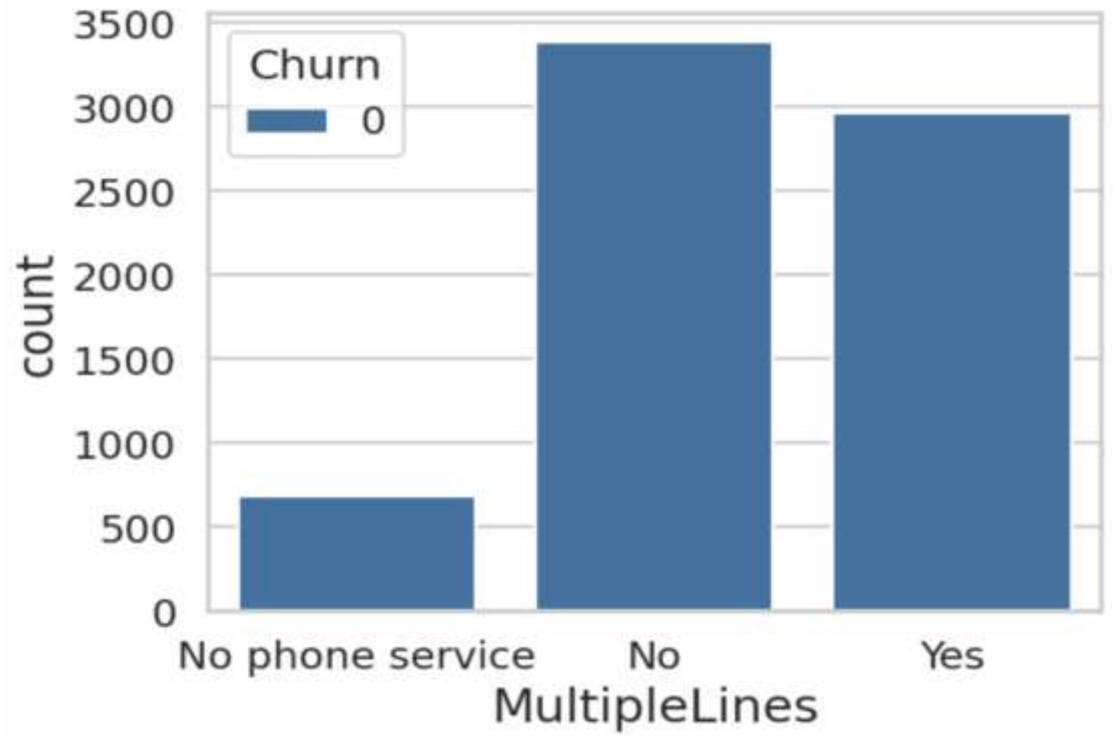
```
[58] for i, predictor in enumerate(telco_data.drop(columns=['Churn', 'TotalCharges', 'MonthlyCharges'])):
         plt.figure(i)
         sns.countplot(data=telco_data, x=predictor, hue='Churn')
```




5000
4000
3000
2000
1000
0
count
Churn
0
No
Yes
Dependents


3500
3000
2500
2000
1500
1000
500
0
count
Churn
0
Yes
No
Partner


Churn
0
3000
2500
2000
1500
1000
500
0
count
DSL
Fiber optic
No
InternetService


Churn
0
3500
3000
2500
2000
1500
1000
500
0
count
No phone service
No
Yes
MultipleLines

2. Convert the target variable 'Churn' in a binary numeric variable i.e. Yes=1 ; No = 0

```
[95] telco_data['Churn'] = np.where(telco_data.Churn == 'Yes',1,0)
```

```
[60] telco_data.head()
```

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Streaming
0	Female	0	Yes	No	No	No phone service	DSL	No	Yes	No	No	N
1	Male	0	No	No	Yes	No	DSL	Yes	No	Yes	No	N
2	Male	0	No	No	Yes	No	DSL	Yes	Yes	No	No	N
3	Male	0	No	No	No	No phone service	DSL	Yes	No	Yes	Yes	N
4	Female	0	No	No	Yes	No	Fiber optic	No	No	No	No	N

3. Convert all the categorical variables into dummy variables

```
[61] telco_data_dummies = pd.get_dummies(telco_data)
     telco_data_dummies.head()
```

	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes	...	PaymentMethod_Bank transfer (automatic)
0	0	29.85	29.85	0	True	False	False	True	True	False	...	False
1	0	56.95	1889.50	0	False	True	True	False	True	False	...	False
2	0	53.85	108.15	0	False	True	True	False	True	False	...	False
3	0	42.30	1840.75	0	False	True	True	False	True	False	...	True
4	0	70.70	151.65	0	True	False	True	False	True	False	...	False

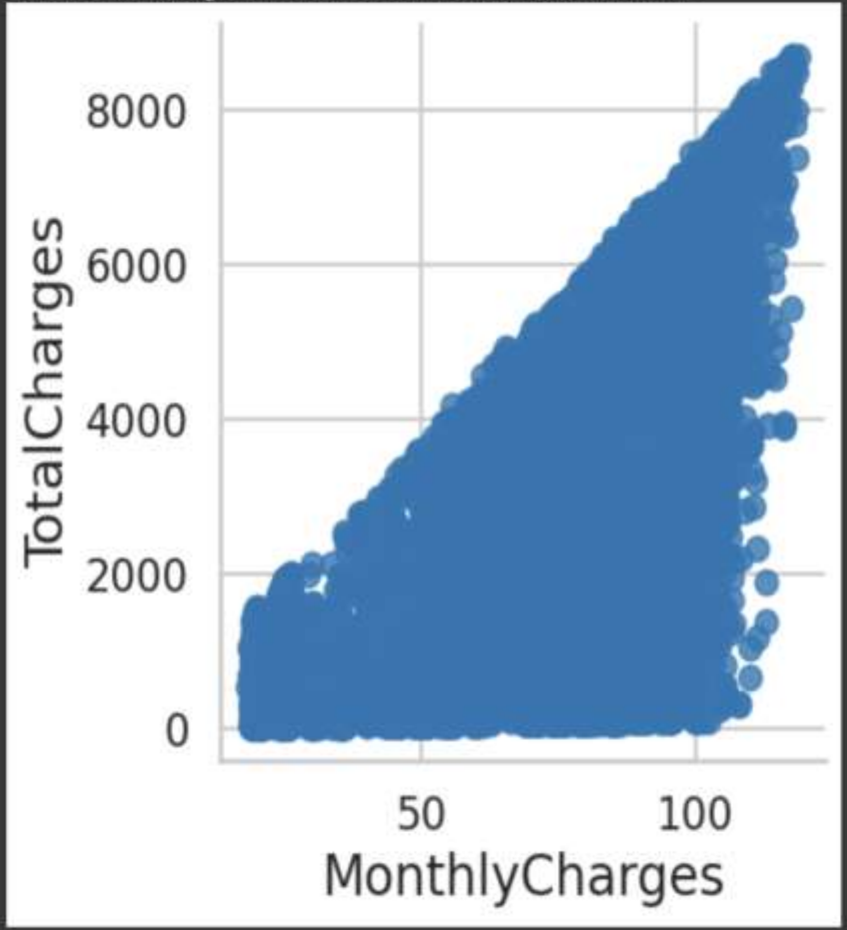
5 rows × 51 columns

SEABORN PLOT

*9. *Relationship between Monthly Charges and Total Charges

```
[62] sns.lmplot(data=telco_data_dummies, x='MonthlyCharges', y='TotalCharges', fit_reg=False)
```

<seaborn.axisgrid.FacetGrid at 0x7f4922dbd330>



*10. *Churn by Monthly Charges and Total Charges

```
[63] Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] == 0) ],
                    color="Red", shade = True)
     Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] == 1) ],
                    ax =Mth, color="Blue", shade= True)
     Mth.legend(["No Churn","Churn"],loc='upper right')
     Mth.set_ylabel('Density')
     Mth.set_xlabel('Monthly Charges')
     Mth.set_title('Monthly charges by churn')
```

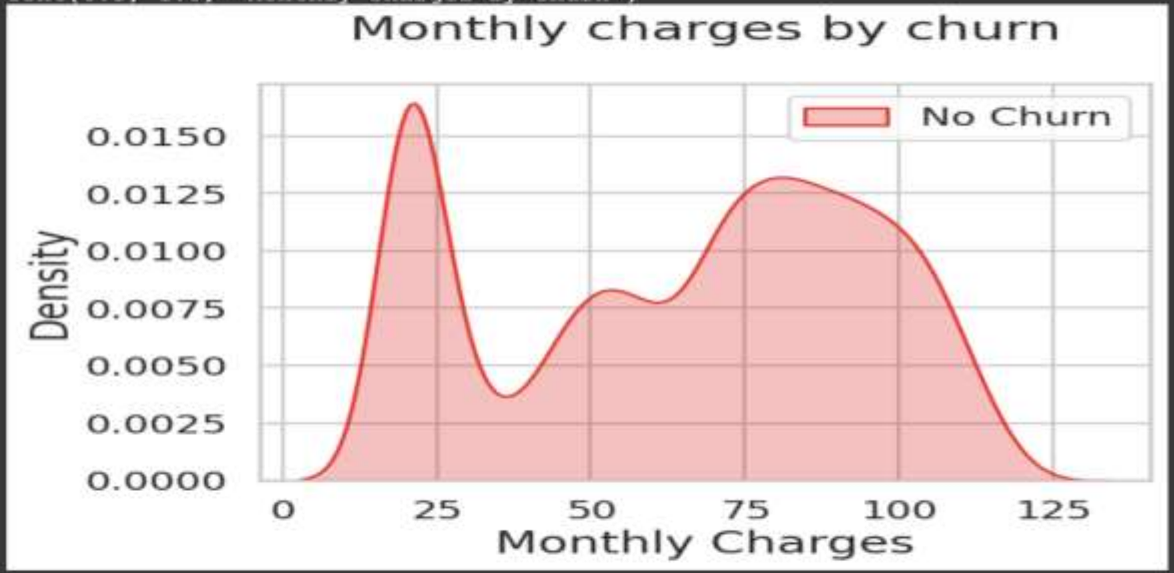
```
<ipython-input-63-940d64c03b8e>:1: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] == 0) ],
<ipython-input-63-940d64c03b8e>:3: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] == 1) ],
Text(0.5, 1.0, 'Monthly charges by churn')
```



Insight: Churn is high when Monthly Charges ar high

```python
Tot = sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Churn"] == 0) ],
                color="Red", shade = True)
Tot = sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Churn"] == 1) ],
                ax =Tot, color="Blue", shade= True)
Tot.legend(["No Churn","Churn"],loc='upper right')
Tot.set_ylabel('Density')
Tot.set_xlabel('Total Charges')
Tot.set_title('Total charges by churn')
```

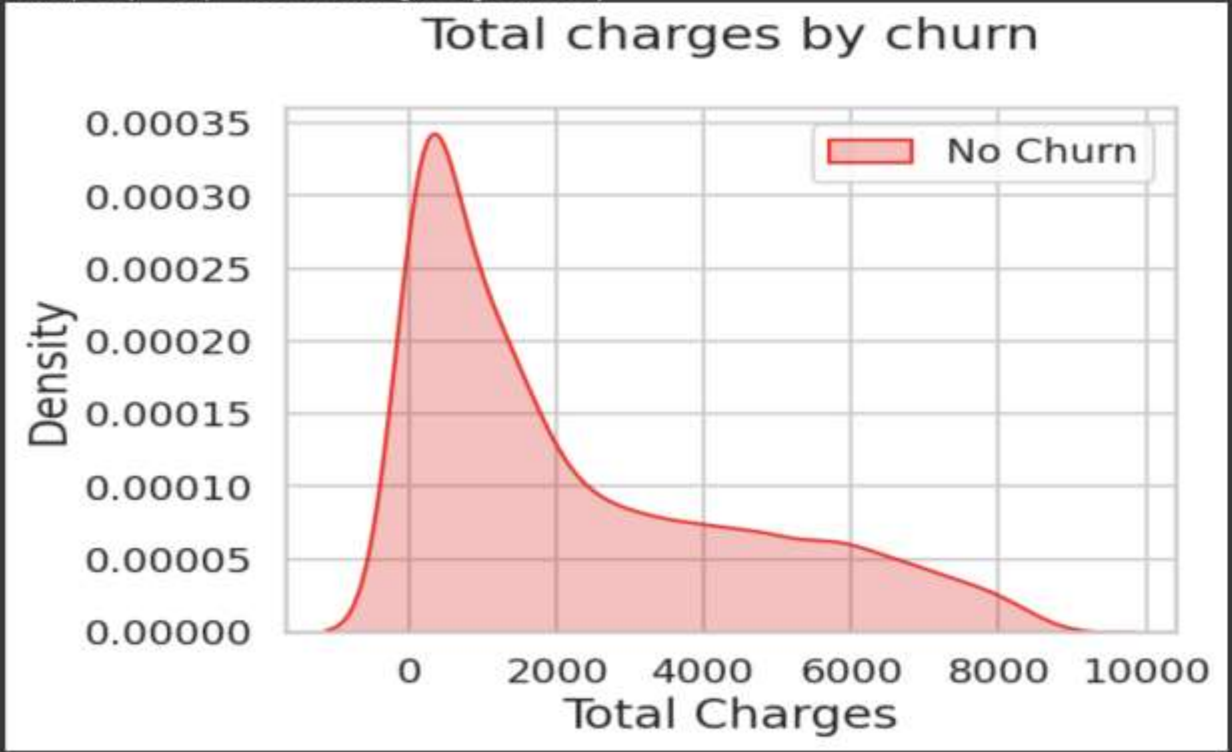
```
<ipython-input-64-aa9d55a4850a>:1: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  Tot = sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Churn"] == 0) ],
<ipython-input-64-aa9d55a4850a>:3: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  Tot = sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Churn"] == 1) ],
Text(0.5, 1.0, 'Total charges by churn')
```



Surprising insight as higher Churn at lower Total Charges

However if we combine the insights of 3 parameters i.e. Tenure, Monthly Charges & Total Charges then the picture is bit clear :- Higher Monthly Charge at lower tenure results into lower Total Charge. Hence, all these 3 factors viz **Higher Monthly Charge**, **Lower tenure** and **Lower Total Charge** are linkd to **High Churn**.

HEATMAP

```
plt.figure(figsize=(12,12))
sns.heatmap(telco_data_dummies.corr(), cmap="Paired")
```

CONCLUSION

- These are some of the quick insights from this exercise:

1. Electronic check medium are the highest churners
2. Contract Type - Monthly customers are more likely to churn because of no contract terms, as they are free to go customers.
3. No Online security, No Tech Support category are high churners
4. Non senior Citizens are high churners

REFERENCES

- Note: There could be many more such insights, so take this as an assignment and try to get more insights :)
- `telco_data_dummies.to_csv('tel_churn.csv')`


THANK
YOU