1.Importing libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

```
df=pd.read_csv('customer_churn_.csv')
df
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...
...	...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...

7043 rows × 21 columns

customerID: A unique identifier for each customer.

gender: The gender of the customer (Male/Female).

SeniorCitizen: Indicates if the customer is a senior citizen (1 = Yes, 0 = No).

Partner: Indicates if the customer has a partner (Yes/No).

Dependents: Indicates if the customer has dependents (Yes/No).

tenure: Number of months the customer has stayed with the company.

PhoneService: Indicates if the customer has a phone service (Yes/No).

MultipleLines: Indicates if the customer has multiple lines (Yes/No/No phone service).

InternetService: Type of internet service (DSL, Fiber optic, None).

OnlineSecurity: Indicates if the customer has online security add-ons (Yes/No/No internet service).

DeviceProtection: Indicates if the customer has device protection add-ons (Yes/No/No internet service).

TechSupport: Indicates if the customer has tech support add-ons (Yes/No/No internet service).

StreamingTV: Indicates if the customer streams TV services (Yes/No/No internet service).

StreamingMovies: Indicates if the customer streams movies (Yes/No/No internet service).

Contract: Type of contract (Month-to-month, One year, Two year).

PaperlessBilling: Indicates if the customer uses paperless billing (Yes/No).

PaymentMethod: The payment method used (e.g., Electronic check, Mailed check, Bank transfer, Credit card).

MonthlyCharges: Monthly charges for the customer.

TotalCharges: Total charges billed to the customer.

Churn: Indicates if the customer has churned (Yes/No).

2.EDA

```
df.isnull().sum().sum()
```

```
np.int64(0)
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

```
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['TotalCharges']=df['TotalCharges'].astype(float) - could not convert string to float
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-8-b9cc9fb2fbbf> in <cell line: 0>()
----> 1 df['TotalCharges']=df['TotalCharges'].astype(float)

                                 6 frames
/usr/local/lib/python3.11/dist-packages/pandas/core/dtypes/astype.py in _astype_nansafe(arr, dtype, copy, skipna)
    131     if copy or arr.dtype == object or dtype == object:
    132         # Explicit copy, or required since NumPy can't view from / to object.
--> 133         return arr.astype(dtype, copy=True)
    134
    135     return arr.astype(dtype, copy=copy)

ValueError: could not convert string to float: ' '
```

Next steps: Explain error

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

```
df.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

```
df.dropna(inplace=True)
df.isnull().sum().sum()
```

```
np.int64(0)
```

```
from sklearn.preprocessing import LabelEncoder

Le=LabelEncoder()
for col in df.columns:
  df[col]=Le.fit_transform(df[col])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7032 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7032 non-null   int64
 1   gender            7032 non-null   int64
 2   SeniorCitizen     7032 non-null   int64
 3   Partner           7032 non-null   int64
 4   Dependents        7032 non-null   int64
 5   tenure            7032 non-null   int64
 6   PhoneService      7032 non-null   int64
 7   MultipleLines     7032 non-null   int64
 8   InternetService   7032 non-null   int64
 9   OnlineSecurity    7032 non-null   int64
 10  OnlineBackup      7032 non-null   int64
 11  DeviceProtection  7032 non-null   int64
 12  TechSupport       7032 non-null   int64
 13  StreamingTV       7032 non-null   int64
 14  StreamingMovies   7032 non-null   int64
 15  Contract          7032 non-null   int64
 16  PaperlessBilling  7032 non-null   int64
 17  PaymentMethod     7032 non-null   int64
 18  MonthlyCharges    7032 non-null   int64
 19  TotalCharges      7032 non-null   int64
 20  Churn             7032 non-null   int64
dtypes: int64(21)
memory usage: 1.2 MB
```

3.Model Building

```
x=df.drop('Churn',axis=1)
y=df['Churn']
```

```
x
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	Onlir
0	5365	0	0	1	0	0	0	1	0	0	
1	3953	1	0	0	0	33	1	0	0	2	
2	2558	1	0	0	0	1	1	0	0	2	
3	5524	1	0	0	0	44	0	1	0	2	
4	6500	0	0	0	0	1	1	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	
7038	4843	1	0	1	1	23	1	2	0	2	
7039	1524	0	0	1	1	71	1	2	1	0	
7040	3358	0	0	1	1	10	0	1	0	2	
7041	5923	1	1	1	0	3	1	2	1	0	
7042	2221	1	0	0	0	65	1	0	1	2	

7032 rows × 20 columns

Next steps: Generate code with x | View recommended plots | New interactive sheet

```
y
```

	Churn
0	0
1	0
2	1
3	0
4	1
...	...
7038	0
7039	0
7040	0
7041	1
7042	0

7032 rows × 1 columns

dtype: int64

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=34)
```

```
model=RandomForestClassifier(n_estimators=50,random_state=23)
```

```
model.fit(x_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(n_estimators=50, random_state=23)
```

```
y_pred=model.predict(x_test)
y_pred
```

```
array([1, 0, 0, ..., 0, 0, 0])
```

```
accuracy_score(y_test,y_pred)*100
```

```
77.82515991471215
```

GridSearchCV

```
# GridSearchCV is a powerful tool in scikit-learn that allows for exhaustive search
# over specified parameter values for an estimator
```

```
from sklearn.model_selection  import GridSearchCV
```

```
base_model=RandomForestClassifier(random_state=23)
```

```
param_grid={
    'n_estimators':[100,200,300],
    'max_depth':[1,5,10],
    'min_samples_split':[2,5,7],
    'min_samples_leaf':[1,2,4],
    'criterion':['gini','entropy']
}
```

```
grid_search=GridSearchCV(estimator=base_model,param_grid=param_grid)
```

```
grid_search.fit(x_train,y_train)
```

```
GridSearchCV
best_estimator_:
RandomForestClassifier
RandomForestClassifier
```

```
print(grid_search.best_params_)
```

```
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 300}
```

```
final_model=RandomForestClassifier(n_estimators=300,max_depth=10,min_samples_leaf=4,min_samples_split=2,criterion= 'entropy',random_state=23)
```

```
final_model.fit(x_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=10, min_samples_leaf=4,
                       n_estimators=300, random_state=23)
```

```
y_pred_final=final_model.predict(x_test)
```

```
accuracy_score(y_test,y_pred_final)*100
```

```
78.96233120113718
```