End-to-End ML Pipeline with Scikit-learn Pipeline API

1. Objective:
   - Build a reusable and production-ready machine learning pipeline for predicting customer churn.
2. Goal:
   - Predict customer churn in the telecom sector using machine learning.
3. Dataset:
   - Telco Customer Churn dataset from IBM (7043 rows, 21 columns).
4. Preprocessing:
   - Handled missing values and incorrect data types.
   - Encoded categorical variables using LabelEncoder.
5. EDA:
   - Plotted churn rates across contracts, genders, services, and payment types.
   - Identified key churn-driving factors.
6. Modeling:
   - Used Logistic Regression and Random Forest with GridSearchCV.
   - Tuned hyperparameters for performance.
7. Evaluation:
   - Used metrics like accuracy, precision, recall, F1-score, and ROC curve.
8. Exported:
   - Final model using joblib.

Section 1: Importing Required Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import missingno as msno
import warnings
warnings.filterwarnings('ignore')
```

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.metrics import roc_curve
from sklearn.metrics import recall_score, confusion_matrix, precision_score, f1_score, accuracy_score, classification_report
import joblib
```

Section 2: Load and Inspect Dataset:

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

```
df.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

5 rows × 21 columns

```
df.shape
```

```
(7043, 21)
```

```
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

```
df.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)
```

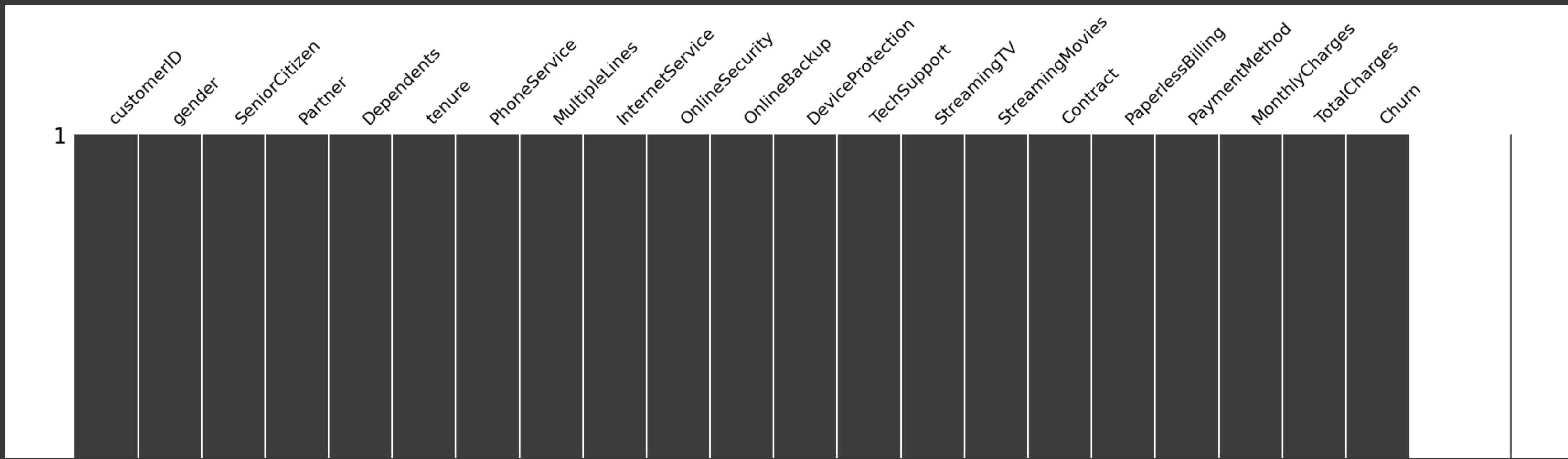
```
df.dtypes
```

	0
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

Section 3: Missing Value Handling & Data Cleaning:

```
msno.matrix(df);
```




```
df = df.drop(['customerID'], axis = 1)
df.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

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

```
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

```
df[np.isnan(df['TotalCharges'])]
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
488	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	No	Yes	Yes	Yes	No	Two year	Yes	Bank transfer (automatic)	52.55	NaN	No
753	Male	0	No	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	No	Mailed check	20.25	NaN	No
936	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	Yes	Yes	No	Yes	Yes	Two year	No	Mailed check	80.85	NaN	No
1082	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	No	Mailed check	25.75	NaN	No
1340	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	Yes	Yes	Yes	Yes	No	Two year	No	Credit card (automatic)	56.05	NaN	No
3331	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	No	Mailed check	19.85	NaN	No
3826	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	No	Mailed check	25.35	NaN	No
4380	Female	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	No	Mailed check	20.00	NaN	No
5218	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	No internet service	No internet service	One year	Yes	Mailed check	19.70	NaN	No
6670	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	Yes	Yes	Yes	Yes	No	Two year	No	Mailed check	73.35	NaN	No
6754	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	Yes	No	Yes	No	No	Two year	Yes	Bank transfer (automatic)	61.90	NaN	No

```
df[df['tenure'] == 0].index
```

```
Index([488, 753, 936, 1082, 1340, 3331, 3826, 4380, 5218, 6670, 6754], dtype='int64')
```

```
df.drop(labels=df[df['tenure'] == 0].index, axis=0, inplace=True)
df[df['tenure'] == 0].index
```

```
Index([], dtype='int64')
```

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

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	No	No	No	One year	No	Mailed check	56.95	1889.50	No
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	No	Yes	Yes	Yes	Yes	One year	Yes	Mailed check	84.80	1990.50	No
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	Yes	Yes	No	Yes	Yes	One year	Yes	Credit card (automatic)	103.20	7362.90	No
7040	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	No	No	No	No	No	Month-to-month	Yes	Electronic check	29.60	346.45	No
7041	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	No	No	No	No	No	Month-to-month	Yes	Mailed check	74.40	306.60	Yes
7042	Male	0	No	No	66	Yes	No	Fiber optic	Yes	No	Yes	Yes	Yes	Yes	Two year	Yes	Bank transfer (automatic)	105.65	6844.50	No

7032 rows × 20 columns

```
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	0
Churn	0

dtype: int64

Section 4: Exploratory Data Analysis (EDA):

```
df["Churn"][df["Churn"]=="No"].groupby(by=df["gender"]).count()
```

gender	Churn
Female	2544
Male	2619

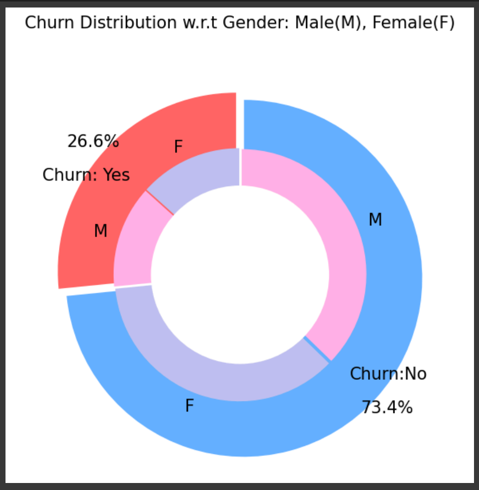
dtype: int64

```
df["Churn"][df["Churn"]=="Yes"].groupby(by=df["gender"]).count()
```

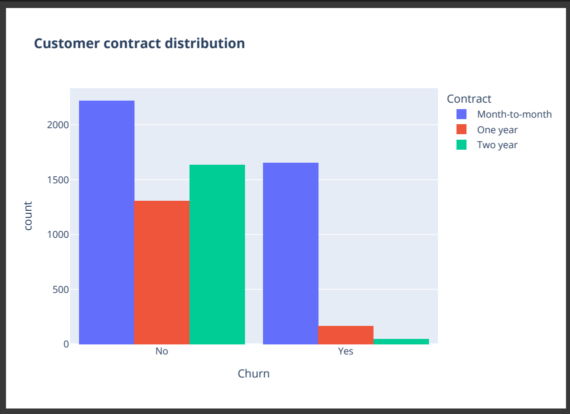
gender	Churn
Female	939
Male	930

dtype: int64

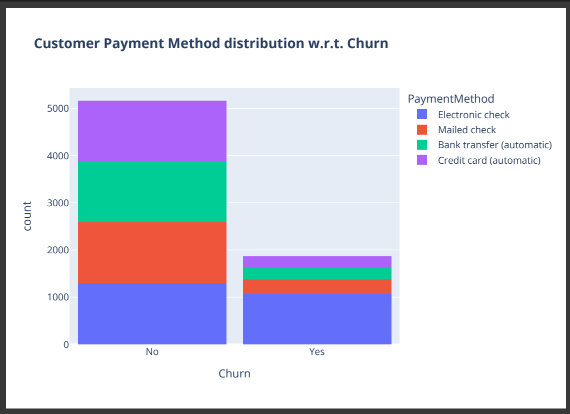
```
plt.figure(figsize=(6, 6))
labels =["Churn: Yes","Churn:No"]
values = [1869,5163]
labels_gender = ["F","M","F","M"]
sizes_gender = [939,930 , 2544,2619]
colors = ['#ff6666', '#66b3ff']
colors_gender = ['#c2c2f0','#ffb3e6', '#c2c2f0','#ffb3e6']
explode = (0.3,0.3)
explode_gender = (0.1,0.1,0.1,0.1)
textprops = {"fontsize":15}
#Plot
plt.pie(values, labels=labels,autopct='%1.1f%%',pctdistance=1.08, labeldistance=0.8,colors=colors, startangle=90,frame=True, explode=explode,radius=10, textprops =textprops, counterclock = True, )
plt.pie(sizes_gender,labels=labels_gender,colors=colors_gender,startangle=90, explode=explode_gender,radius=7, textprops =textprops, counterclock = True, )
#Draw circle
centre_circle = plt.Circle((0,0),5,color='black', fc='white',linewidth=0)
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.title('Churn Distribution w.r.t Gender: Male(M), Female(F)', fontsize=15, y=1.1)

# show plot

plt.axis('equal')
plt.tight_layout()
plt.show()
```




```
fig = px.histogram(df, x="Churn", color="Contract", barmode="group", title="<b>Customer contract distribution<b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```




```
fig = px.histogram(df, x="Churn", color="PaymentMethod", title="<b>Customer Payment Method distribution w.r.t. Churn</b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```




```
df["InternetService"].unique()
```

```
array(['DSL', 'Fiber optic', 'No'], dtype=object)
```

```
df[df["gender"]=="Male"][["InternetService", "Churn"]].value_counts()
```

InternetService	Churn	count
DSL	No	992
Fiber optic	No	910
No	No	717
Fiber optic	Yes	633
DSL	Yes	240
No	Yes	57

dtype: int64

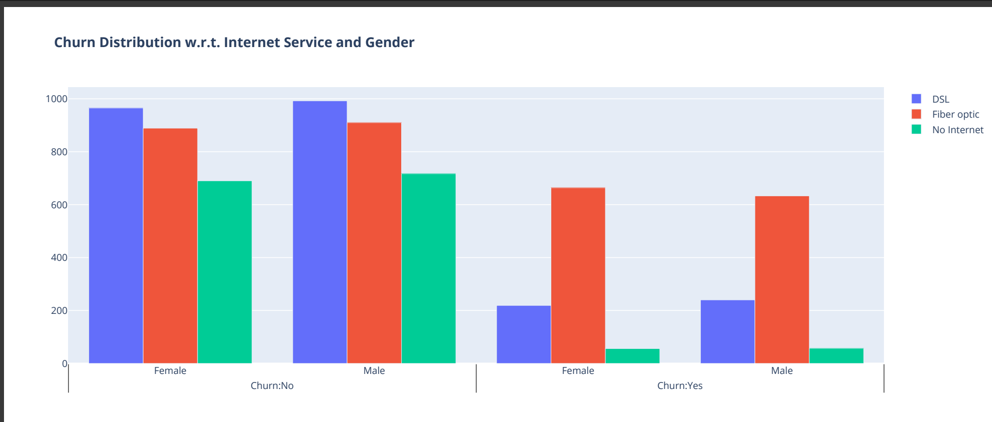
```
df[df["gender"]=="Female"][["InternetService", "Churn"]].value_counts()
```

InternetService	Churn	count
DSL	No	965
Fiber optic	No	889
No	No	690
Fiber optic	Yes	664
DSL	Yes	219
No	Yes	56

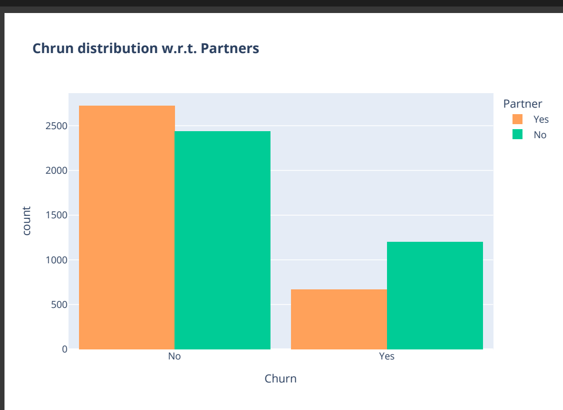
dtype: int64

```
fig = go.Figure()

fig.add_trace(go.Bar(
  x = [['Churn:No', 'Churn:No', 'Churn:Yes', 'Churn:Yes'],
       ["Female", "Male", "Female", "Male"]],
  y = [965, 992, 219, 240],
  name = 'DSL',
))

fig.add_trace(go.Bar(
  x = [['Churn:No', 'Churn:No', 'Churn:Yes', 'Churn:Yes'],
       ["Female", "Male", "Female", "Male"]],
  y = [889, 910, 664, 633],
  name = 'Fiber optic',
))

fig.add_trace(go.Bar(
  x = [['Churn:No', 'Churn:No', 'Churn:Yes', 'Churn:Yes'],
       ["Female", "Male", "Female", "Male"]],
  y = [690, 717, 56, 57],
  name = 'No Internet',
))
```

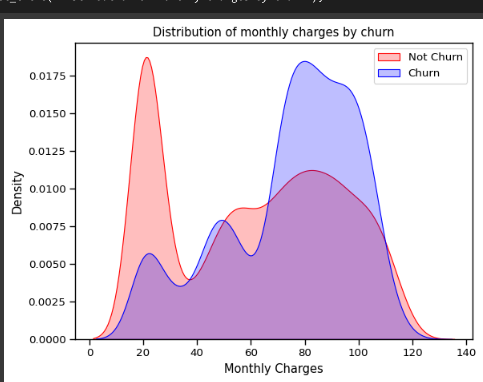
```
fig.update_layout(title_text="<b>Churn Distribution w.r.t. Internet Service and Gender</b>")

fig.show()
```



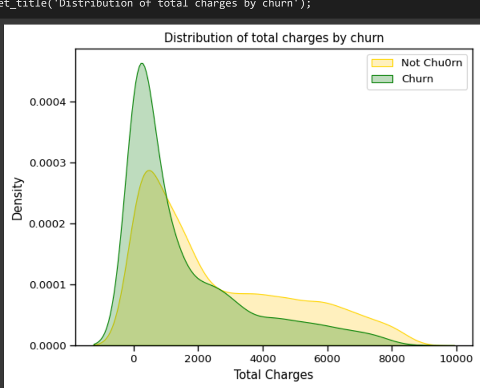

```
color_map = {"Yes": '#FFA15A', "No": '#00CC96'}
fig = px.histogram(df, x="Churn", color="Partner", barmode="group", title="<b>Chrun distribution w.r.t. Partners</b>", color_discrete_map=color_map)
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```



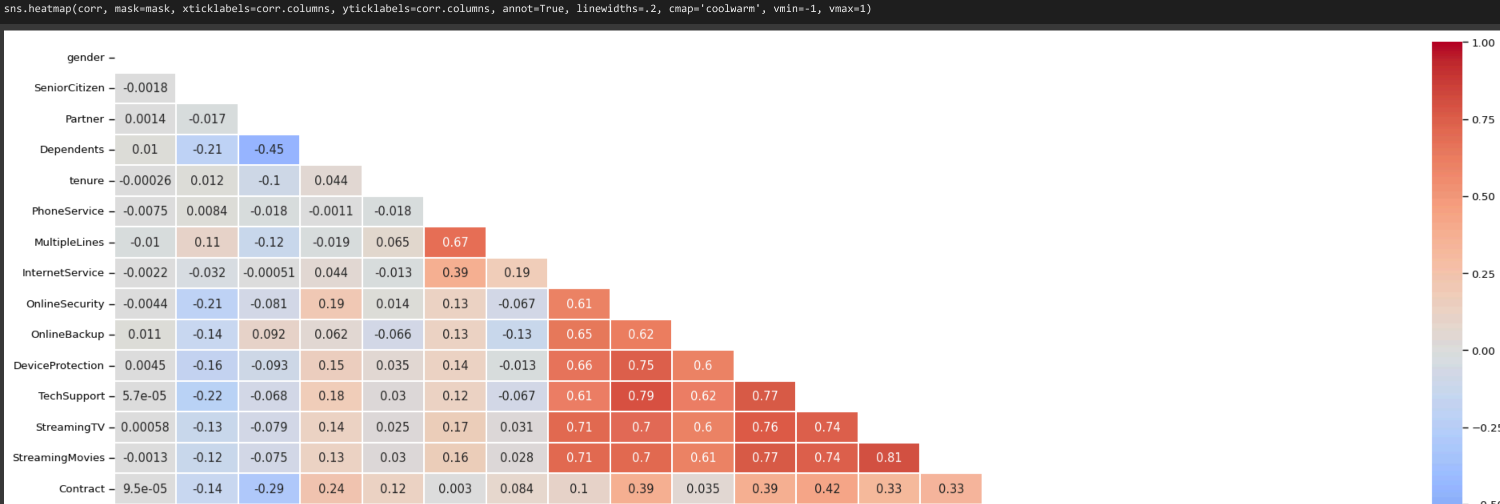

```
sns.set_context("paper",font_scale=1.1)
ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'No') ],
                color="Red", shade = True);
ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'Yes') ],
                ax =ax, color="Blue", shade= True);
ax.legend(["Not Churn","Churn"],loc='upper right');
ax.set_ylabel('Density');
ax.set_xlabel('Monthly Charges');
ax.set_title('Distribution of monthly charges by churn');
```




```
ax = sns.kdeplot(df.TotalCharges[(df["Churn"] == 'No') ],
                color="Gold", shade = True);
ax = sns.kdeplot(df.TotalCharges[(df["Churn"] == 'Yes') ],
                ax =ax, color="Green", shade= True);
ax.legend(["Not Chu0rn","Churn"],loc='upper right');
ax.set_ylabel('Density');
ax.set_xlabel('Total Charges');
ax.set_title('Distribution of total charges by churn');
```




```
plt.figure(figsize=(25, 10))

corr = df.apply(lambda x: pd.factorize(x)[0]).corr()

mask = np.triu(np.ones_like(corr, dtype=bool))

ax = sns.heatmap(corr, mask=mask, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, linewidths=.2, cmap='coolwarm', vmin=-1, vmax=1)
```



Section 5: Label Encoding for Modeling:

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

```
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	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	0	0	1	0	1	0	1	0	0	2	0	0	0	0	0	1	2	29.85	29.85	0
1	1	0	0	0	34	1	0	0	2	0	2	0	0	0	1	0	3	56.95	1889.50	0
2	1	0	0	0	2	1	0	0	2	2	0	0	0	0	0	1	3	53.85	108.15	1
3	1	0	0	0	45	0	1	0	2	0	2	2	0	0	1	0	0	42.30	1840.75	0
4	0	0	0	0	2	1	0	1	0	0	0	0	0	0	0	1	2	70.70	151.65	1

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

```
plt.figure(figsize=(14,7))
df.corr()['Churn'].sort_values(ascending = False)
```

	Churn
Churn	1.000000
MonthlyCharges	0.192858
PaperlessBilling	0.191454
SeniorCitizen	0.150541
PaymentMethod	0.107852
MultipleLines	0.038043
PhoneService	0.011691
gender	-0.008545
StreamingTV	-0.036303
StreamingMovies	-0.038802
InternetService	-0.047097
Partner	-0.149982
Dependents	-0.163128
DeviceProtection	-0.177883
OnlineBackup	-0.195290
TotalCharges	-0.199484
TechSupport	-0.282232
OnlineSecurity	-0.289050
tenure	-0.354049
Contract	-0.396150

dtype: float64
```
<Figure size 1400x700 with 0 Axes>
```

Section 6: Feature Selection & Splitting:

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

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.30, random_state = 40, stratify=y)
```

Section 7: Building Pipelines and Hyperparameter Tuning:

```
# Logistic Regression Pipeline
logreg_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('logreg', LogisticRegression(max_iter=1000, random_state=42))
])

# Random Forest Pipeline
rf_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('rf', RandomForestClassifier(random_state=42))
])
```

```
from sklearn.model_selection import GridSearchCV

# Hyperparameter grids
logreg_param_grid = {
    'logreg__C': [0.01, 0.1, 1.0, 10]
}

rf_param_grid = {
    'rf__n_estimators': [50, 100],
    'rf__max_depth': [5, 10, None]
}
```

```
# Logistic Regression Grid Search
logreg_grid = GridSearchCV(logreg_pipeline, logreg_param_grid, cv=5, n_jobs=-1)
logreg_grid.fit(X_train, y_train)

# Random Forest Grid Search
rf_grid = GridSearchCV(rf_pipeline, rf_param_grid, cv=5, n_jobs=-1)
rf_grid.fit(X_train, y_train)
```

```
GridSearchCV
best_estimator_: Pipeline
StandardScaler
RandomForestClassifier
```

Section 8: Model Evaluation:

```
# Evaluate Logistic Regression
logreg_preds = logreg_grid.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, logreg_preds))
print("Confusion Matrix:\n", confusion_matrix(y_test, logreg_preds))
print("Classification Report:\n", classification_report(y_test, logreg_preds))

# Evaluate Random Forest
rf_preds = rf_grid.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, rf_preds))
print("Confusion Matrix:\n", confusion_matrix(y_test, rf_preds))
print("Classification Report:\n", classification_report(y_test, rf_preds))
```

```
Logistic Regression Accuracy: 0.8075829383886256
Confusion Matrix:
 [[1384  165]
 [ 241  320]]
Classification Report:
               precision    recall  f1-score   support

           0       0.85      0.89      0.87      1549
           1       0.66      0.57      0.61       561

    accuracy                           0.81      2110
   macro avg       0.76      0.73      0.74      2110
weighted avg       0.80      0.81      0.80      2110

Random Forest Accuracy: 0.809478672985782
Confusion Matrix:
 [[1406  143]
 [ 259  302]]
Classification Report:
               precision    recall  f1-score   support

           0       0.84      0.91      0.87      1549
           1       0.68      0.54      0.60       561

    accuracy                           0.81      2110
   macro avg       0.76      0.72      0.74      2110
weighted avg       0.80      0.81      0.80      2110
```

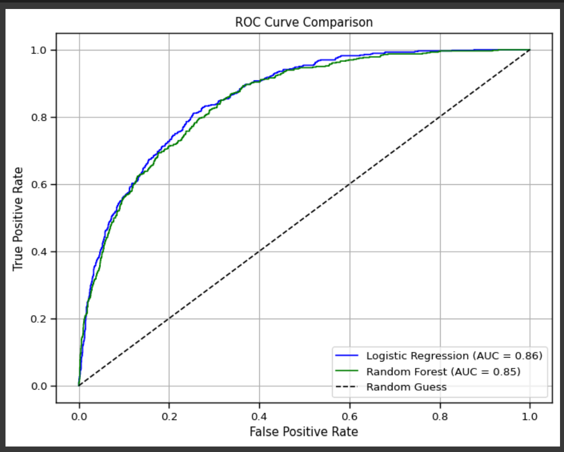
```
from sklearn.metrics import roc_curve, roc_auc_score
```

```
logreg_proba = logreg_grid.predict_proba(X_test)[:, 1]
rf_proba = rf_grid.predict_proba(X_test)[:, 1]
```

```
fpr_log, tpr_log, _ = roc_curve(y_test, logreg_proba)
fpr_rf, tpr_rf, _ = roc_curve(y_test, rf_proba)
```

```
auc_log = roc_auc_score(y_test, logreg_proba)
auc_rf = roc_auc_score(y_test, rf_proba)
```

```
plt.figure(figsize=(8, 6))
plt.plot(fpr_log, tpr_log, label=f'Logistic Regression (AUC = {auc_log:.2f})', color='blue')
plt.plot(fpr_rf, tpr_rf, label=f'Random Forest (AUC = {auc_rf:.2f})', color='green')
plt.plot([0, 1], [0, 1], 'k--', label='Random Guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```



Section 9: Model Saving:

```
import joblib

# Save the best models
joblib.dump(logreg_grid.best_estimator_, "best_logistic_model_pipeline.pkl")
joblib.dump(rf_grid.best_estimator_, "best_rf_model_pipeline.pkl")
```

```
['best_rf_model_pipeline.pkl']
```