```python
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

```python
tf.config.list_physical_devices('GPU')
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```python
import kagglehub

# Download latest version
path = kagglehub.dataset_download("blastchar/telco-customer-churn")

print("Path to dataset files:", path)
```

```
Path to dataset files: /root/.cache/kagglehub/datasets/blastchar/telco-customer-churn/versions/1
```

```python
path
```

```python
import os

filepath = os.path.join(path, "WA_Fn-UseC_-Telco-Customer-Churn.csv")
df = pd.read_csv(filepath)
print(df.shape)
df.head()
```

(7043, 21)

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

5 rows × 21 columns

Cleaning Data

```
# Drop unimportant columns

df.drop('customerID', axis=1, inplace=True)


# Check null values
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

```
# Check duplicate values
df.duplicated().sum()
```

22

```
df.drop_duplicates(inplace=True, keep='first')
```

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

0

```
df.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSec
0	Female	0	Yes	No	1	No	No phone service	DSL	
1	Male	0	No	No	34	Yes	No	DSL	
2	Male	0	No	No	2	Yes	No	DSL	
3	Male	0	No	No	45	No	No phone service	DSL	
4	Female	0	No	No	2	Yes	No	Fiber optic	

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7021 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   gender            7021 non-null   object 
 1   SeniorCitizen     7021 non-null   int64  
 2   Partner           7021 non-null   object 
 3   Dependents        7021 non-null   object 
 4   tenure            7021 non-null   int64  
 5   PhoneService      7021 non-null   object 
 6   MultipleLines     7021 non-null   object 
 7   InternetService   7021 non-null   object 
 8   OnlineSecurity    7021 non-null   object 
 9   OnlineBackup      7021 non-null   object 
 10  DeviceProtection  7021 non-null   object 
 11  TechSupport       7021 non-null   object 
 12  StreamingTV       7021 non-null   object 
 13  StreamingMovies   7021 non-null   object 
 14  Contract          7021 non-null   object 
 15  PaperlessBilling  7021 non-null   object 
 16  PaymentMethod     7021 non-null   object 
 17  MonthlyCharges    7021 non-null   float64
 18  TotalCharges      7021 non-null   object 
 19  Churn             7021 non-null   object 
dtypes: float64(1), int64(2), object(17)
memory usage: 1.1+ MB
```

```
df[df['TotalCharges']==" "]
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	Online
488	Female	0	Yes	Yes	0	No	No phone service	DSL	
753	Male	0	No	Yes	0	Yes	No	No	No inter
936	Female	0	Yes	Yes	0	Yes	No	DSL	
1082	Male	0	Yes	Yes	0	Yes	Yes	No	No inter
1340	Female	0	Yes	Yes	0	No	No phone service	DSL	
3331	Male	0	Yes	Yes	0	Yes	No	No	No inter
3826	Male	0	Yes	Yes	0	Yes	Yes	No	No inter
4380	Female	0	Yes	Yes	0	Yes	No	No	No inter
5218	Male	0	Yes	Yes	0	Yes	No	No	No inter
6670	Female	0	Yes	Yes	0	Yes	Yes	DSL	
6754	Male	0	No	Yes	0	Yes	Yes	DSL	

```
df[df['TotalCharges']==" "].shape
```

```
(11, 20)
```

```
df = df[df['TotalCharges']!=" "]
```

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

```
<ipython-input-135-1b11d6a2f456>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
  df['TotalCharges'] = pd.to_numeric(df['TotalCharges'])
```

```
df['TotalCharges'].dtype
```

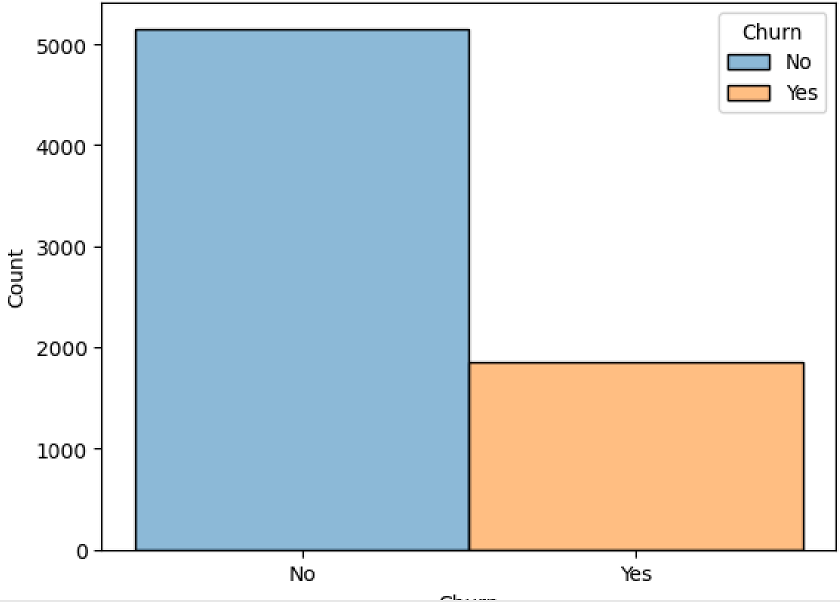
```
dtype('float64')
```

```
df.shape
```

```
(7010, 20)
```

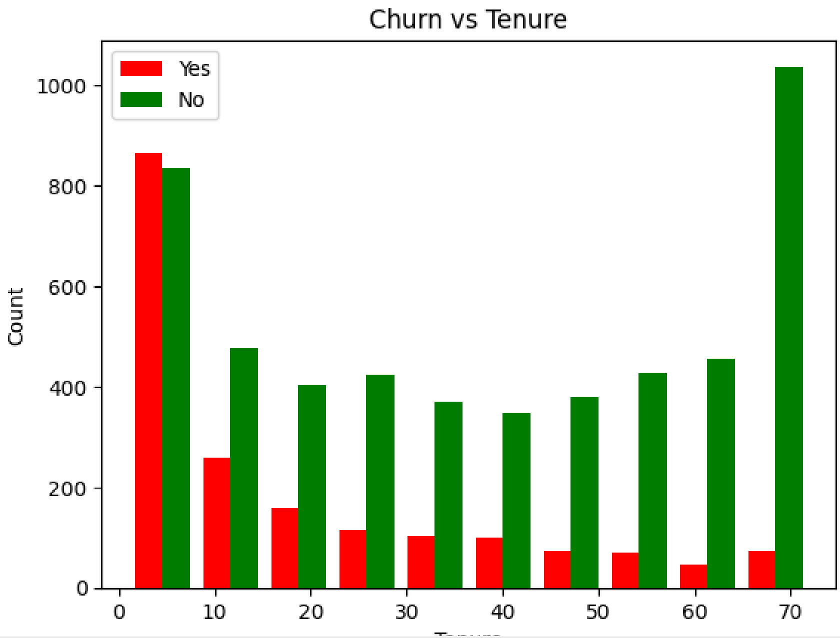
```
sns.histplot(df, x = 'Churn', hue='Churn')
```

```
<Axes: xlabel='Churn', ylabel='Count'>
```



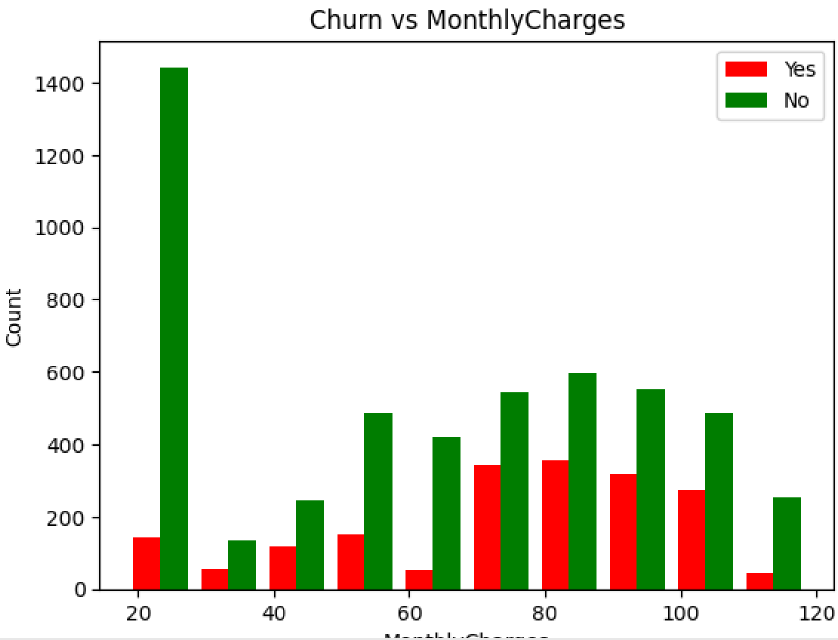

```
cm_yes_churn_tennure = df[df['Churn']=='Yes']['tenure']
cm_no_churn_tennure = df[df['Churn']=='No']['tenure']

# blood_sugar_men = [113, 85, 90, 150, 149, 88, 93, 115, 135, 80, 77, 82, 129]
# blood_sugar_women = [67, 98, 89, 120, 133, 150, 84, 69, 89, 79, 120, 112, 100]
plt.hist([cm_yes_churn_tennure, cm_no_churn_tennure], color=['red', 'green'], label=['Yes', 'No'])
plt.legend()
plt.xlabel('Tenure')
```

```
plt.ylabel('Count')
plt.title('Churn vs Tenure')
plt.show()
```




```
churn_yes_MonthlyCharges = df[df['Churn']=='Yes']['MonthlyCharges']
churn_no_MonthlyCharges = df[df['Churn']=='No']['MonthlyCharges']
plt.hist([churn_yes_MonthlyCharges, churn_no_MonthlyCharges], color=['red', 'green'], label=['Yes', 'No'])
```

```
plt.legend()
plt.xlabel('MonthlyCharges')
plt.ylabel('Count')
plt.title('Churn vs MonthlyCharges')
plt.show()
```



Check Spelling

```
for col in df.select_dtypes(include=('object')):
    print(f'{col}: {df[col].unique()}')
```

```
gender: ['Female' 'Male']
Partner: ['Yes' 'No']
Dependents: ['No' 'Yes']
PhoneService: ['No' 'Yes']
MultipleLines: ['No phone service' 'No' 'Yes']
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: ['No' 'Yes' 'No internet service']
OnlineBackup: ['Yes' 'No' 'No internet service']
DeviceProtection: ['No' 'Yes' 'No internet service']
TechSupport: ['No' 'Yes' 'No internet service']
StreamingTV: ['No' 'Yes' 'No internet service']
StreamingMovies: ['No' 'Yes' 'No internet service']
Contract: ['Month-to-month' 'One year' 'Two year']
PaperlessBilling: ['Yes' 'No']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn: ['No' 'Yes']
```

```
df['MultipleLines'].replace('No phone service', 'No', inplace=True)
df['OnlineSecurity'].replace('No internet service', 'No', inplace=True)
df['OnlineBackup'].replace('No internet service', 'No', inplace=True)
df['DeviceProtection'].replace('No internet service', 'No', inplace=True)
df['TechSupport'].replace('No internet service', 'No', inplace=True)
df['StreamingTV'].replace('No internet service', 'No', inplace=True)
df['StreamingMovies'].replace('No internet service', 'No', inplace=True)
```

Show hidden output

```
for col in df.select_dtypes(include=('object')):
    print(f'{col}: {df[col].unique()}')
```

```
gender: ['Female' 'Male']
Partner: ['Yes' 'No']
Dependents: ['No' 'Yes']
PhoneService: ['No' 'Yes']
MultipleLines: ['No' 'Yes']
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: ['No' 'Yes']
OnlineBackup: ['Yes' 'No']
DeviceProtection: ['No' 'Yes']
TechSupport: ['No' 'Yes']
StreamingTV: ['No' 'Yes']
StreamingMovies: ['No' 'Yes']
Contract: ['Month-to-month' 'One year' 'Two year']
PaperlessBilling: ['Yes' 'No']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn: ['No' 'Yes']
```

```
len(df['gender'].unique())
```

```
2
```

```
holder_column = []
for value in df.select_dtypes(include=('object')):
    if len(df[value].unique()) == 2:
        holder_column.append(value)
```

```
holder_column
```

```
['gender',
 'Partner',
 'Dependents',
 'PhoneService',
 'MultipleLines',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'PaperlessBilling',
 'Churn']
```

```
list_encoder = ['Partner',
 'Dependents',
 'PhoneService',
 'MultipleLines',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'PaperlessBilling',
 'Churn']
```

```
for col in list_encoder:
    df[col] = df[col].map({'Yes': 1, 'No': 0})
```

```
for col in df.select_dtypes(include=('object')):
    print(f'{col}: {df[col].unique()}')
```

```
gender: ['Female' 'Male']
InternetService: ['DSL' 'Fiber optic' 'No']
Contract: ['Month-to-month' 'One year' 'Two year']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
```

```
list_categories = ['gender', 'InternetService', 'Contract', 'PaymentMethod']

df = pd.get_dummies( df, columns =list_categories, drop_first=True, dtype=int)
```

```
df.head()
```

	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	Devi
0	0	1	0	1	0	0	0	1	
1	0	0	0	34	1	0	1	0	
2	0	0	0	2	1	0	1	1	
3	0	0	0	45	0	0	1	0	
4	0	0	0	2	1	0	0	0	

5 rows × 24 columns

```
df.shape
```

```
(7010, 24)
```

```
for col in df.columns:
    print(f'{col}: {df[col].unique()}')
```

```
SeniorCitizen: [0 1]
Partner: [1 0]
Dependents: [0 1]
tenure: [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]
PhoneService: [0 1]
MultipleLines: [0 1]
OnlineSecurity: [0 1]
OnlineBackup: [1 0]
DeviceProtection: [0 1]
TechSupport: [0 1]
StreamingTV: [0 1]
StreamingMovies: [0 1]
PaperlessBilling: [1 0]
MonthlyCharges: [29.85 56.95 53.85 ... 63.1  44.2  78.7 ]
TotalCharges: [  29.85 1889.5   108.15 ...  346.45  306.6  6844.5 ]
Churn: [0 1]
gender_Male: [0 1]
InternetService_Fiber optic: [0 1]
InternetService_No: [0 1]
Contract_One year: [0 1]
Contract_Two year: [0 1]
PaymentMethod_Credit card (automatic): [0 1]
PaymentMethod_Electronic check: [1 0]
PaymentMethod_Mailed check: [0 1]
```

```
list_numerical = ['tenure', 'MonthlyCharges', 'TotalCharges']
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[list_numerical] = scaler.fit_transform(df[list_numerical])
```

```
for col in df.columns:
    print(f'{col}: {df[col].unique()}')
```

```
SeniorCitizen: [0 1]
Partner: [1 0]
Dependents: [0 1]
tenure: [0.         0.46478873 0.01408451 0.61971831 0.09859155 0.29577465
 0.12676056 0.38028169 0.85915493 0.16901408 0.21126761 0.8028169
 0.67605634 0.33802817 0.95774648 0.71830986 0.98591549 0.28169014
 0.15492958 0.4084507  0.64788732 1.         0.22535211 0.36619718
 0.05633803 0.63380282 0.14084507 0.97183099 0.87323944 0.5915493
 0.1971831  0.83098592 0.23943662 0.91549296 0.11267606 0.02816901
 0.42253521 0.69014085 0.88732394 0.77464789 0.08450704 0.57746479
 0.47887324 0.66197183 0.3943662  0.90140845 0.52112676 0.94366197
 0.43661972 0.76056338 0.50704225 0.49295775 0.56338028 0.07042254
 0.04225352 0.45070423 0.92957746 0.30985915 0.78873239 0.84507042
 0.18309859 0.26760563 0.73239437 0.54929577 0.81690141 0.32394366
 0.6056338  0.25352113 0.74647887 0.70422535 0.35211268 0.53521127]
PhoneService: [0 1]
MultipleLines: [0 1]
OnlineSecurity: [0 1]
OnlineBackup: [1 0]
DeviceProtection: [0 1]
TechSupport: [0 1]
StreamingTV: [0 1]
StreamingMovies: [0 1]
PaperlessBilling: [1 0]
MonthlyCharges: [0.11542289 0.38507463 0.35422886 ... 0.44626866 0.25820896 0.60149254]
TotalCharges: [0.0012751  0.21586661 0.01031041 ... 0.03780868 0.03321025 0.78764136]
Churn: [0 1]
gender_Male: [0 1]
InternetService_Fiber optic: [0 1]
InternetService_No: [0 1]
Contract_One year: [0 1]
Contract_Two year: [0 1]
PaymentMethod_Credit card (automatic): [0 1]
```

```
PaymentMethod_Electronic check: [1 0]
PaymentMethod_Mailed check: [0 1]
```

Double-click (or enter) to edit

```
# Check VIF or Correlation
```



```
from statsmodels.stats.outliers_influence import variance_inflation_factor

def check_vif(data):
    df = pd.DataFrame()
    df["features"] = data.columns
    df["VIF"] = [variance_inflation_factor(data.values, i) for i in range(data.shape[1])]
    return df

df_vif = check_vif(df.drop('Churn', axis=1))
df_vif
```

	features	VIF
0	SeniorCitizen	1.373530
1	Partner	2.820676
2	Dependents	1.968703
3	tenure	18.493939
4	PhoneService	184.809375
5	MultipleLines	6.107642
6	OnlineSecurity	5.365767
7	OnlineBackup	6.027084
8	DeviceProtection	6.098552
9	TechSupport	5.452006
10	StreamingTV	17.794517
11	StreamingMovies	18.208771
12	PaperlessBilling	2.895726
13	MonthlyCharges	1108.782774
14	TotalCharges	19.893650
15	gender_Male	1.994984
16	InternetService_Fiber optic	107.432340
17	InternetService_No	49.737430
18	Contract_One year	2.059725
19	Contract_Two year	3.488994

20	PaymentMethod_Credit card (automatic)	1.926215
21	PaymentMethod_Electronic check	2.799744

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

Generated code may be subject to a license | MaxLukinDS/Marketing-Analytics-with-Python |

```
df_vif[df_vif['VIF']>20].sort_values(by='VIF', ascending=False).features.tolist()
```

```
['MonthlyCharges',
 'PhoneService',
 'InternetService_Fiber optic',
 'InternetService_No']
```

```
df.drop(df_vif[df_vif['VIF']>20].sort_values(by='VIF', ascending=False).features.tolist(), axis=1, inplace=True)
```

```
df_vif = check_vif(df.drop('Churn', axis=1))
df_vif
```

	features	VIF
0	SeniorCitizen	1.345035
1	Partner	2.801339
2	Dependents	1.945300
3	tenure	13.968483
4	MultipleLines	2.304620
5	OnlineSecurity	1.812154
6	OnlineBackup	2.122822
7	DeviceProtection	2.256101
8	TechSupport	1.950543
9	StreamingTV	2.742954
10	StreamingMovies	2.765361
11	PaperlessBilling	2.548340
12	TotalCharges	13.799645
13	gender_Male	1.865646
14	Contract_One year	1.936879
15	Contract_Two year	3.196916
16	PaymentMethod_Credit card (automatic)	1.640066
17	PaymentMethod_Electronic check	2.034507
18	PaymentMethod_Mailed check	1.460859

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

```
df_vif[df_vif['VIF']>10].sort_values(by='VIF', ascending=False).features.tolist()
```

```
['tenure', 'TotalCharges']
```

```
df.drop(df_vif[df_vif['VIF']>10].sort_values(by='VIF', ascending=False).features.tolist(), axis=1, inplace=True
```

```
df_vif = check_vif(df.drop('Churn', axis=1))
df_vif
```

	features	VIF
0	SeniorCitizen	1.336754
1	Partner	2.659412
2	Dependents	1.931751
3	MultipleLines	1.988821
4	OnlineSecurity	1.735996
5	OnlineBackup	1.910806
6	DeviceProtection	2.142202
7	TechSupport	1.906247
8	StreamingTV	2.552191
9	StreamingMovies	2.582269
10	PaperlessBilling	2.514802
11	gender_Male	1.839142
12	Contract_One year	1.505680
13	Contract_Two year	1.880157
14	PaymentMethod_Credit card (automatic)	1.597696
15	PaymentMethod_Electronic check	1.965422
16	PaymentMethod_Mailed check	1.394937

Next steps: Generate code with `df_vif` View recommended plots New interactive sheet

```
print(df.shape)
df.head()
```

(7010, 18)

	SeniorCitizen	Partner	Dependents	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSup
0	0	1	0	0	0	1	0	
1	0	0	0	0	1	0	1	
2	0	0	0	0	1	1	0	
3	0	0	0	0	1	0	1	
4	0	0	0	0	0	0	0	

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

Training model

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

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
model  = keras.Sequential([
    keras.layers.Dense(128, input_shape=(17,), activation='relu'),
```

```
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
with tf.device('/GPU:0'):
    model.fit(X_train, y_train, epochs=100)
```

```
Epoch 1/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7839 - loss: 0.4308
Epoch 2/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7802 - loss: 0.4291
Epoch 3/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7805 - loss: 0.4325
Epoch 4/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7887 - loss: 0.4236
Epoch 5/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7969 - loss: 0.4218
Epoch 6/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7789 - loss: 0.4306
Epoch 7/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.7930 - loss: 0.4160
Epoch 8/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7994 - loss: 0.4107
Epoch 9/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.7921 - loss: 0.4029
Epoch 10/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7976 - loss: 0.4037
```

```
Epoch 11/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7973 - loss: 0.4050
Epoch 12/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8017 - loss: 0.3948
Epoch 13/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8061 - loss: 0.3878
Epoch 14/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8103 - loss: 0.3824
Epoch 15/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8091 - loss: 0.3790
Epoch 16/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8249 - loss: 0.3619
Epoch 17/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8239 - loss: 0.3642
Epoch 18/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8237 - loss: 0.3672
Epoch 19/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8260 - loss: 0.3580
Epoch 20/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8302 - loss: 0.3481
Epoch 21/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8350 - loss: 0.3405
Epoch 22/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8308 - loss: 0.3437
Epoch 23/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8351 - loss: 0.3310
Epoch 24/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8346 - loss: 0.3409
Epoch 25/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8429 - loss: 0.3173
Epoch 26/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8398 - loss: 0.3314
Epoch 27/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8573 - loss: 0.3124
Epoch 28/100
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8500 - loss: 0.3049
Epoch 29/100
```

```
model.evaluate(X_test, y_test)
```

```
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7661 - loss: 0.9286
[0.9089354872703552, 0.7689015865325928]
```

```
from sklearn.metrics import confusion_matrix, classification_report
y_pred = model.predict(X_test)
y_pred[:5]
```

```
44/44 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
array([[3.0730885e-06],
       [2.1227651e-12],
       [7.6985466e-01],
       [1.7375077e-01],
       [7.7014347e-04]], dtype=float32)
```

```
y_predict = []
for i in range(len(y_pred)):
    if y_pred[i] >= 0.5:
       y_predict.append(1)
    else:
        y_predict.append(0)
```

```
y_predict[:10]
```

```
[0, 0, 1, 0, 0, 1, 1, 0, 0, 0]
```