LOAD and CLEAN DATA

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv('Telecom_Customer_Churn.csv')
df['total_charges'] = pd.to_numeric(df['total_charges'], errors = 'coerce')
df = df. dropna(subset=['total_charges'])
df.head()
```

Out[2]:

	customer_id	gender	senior_citizen	partner	dependents	tenure	phone_service	multip
0	7590-VHVEG	Female	0	Yes	No	1	No	N
1	5575-GNVDE	Male	0	No	No	34	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	
3	7795-CFOCW	Male	0	No	No	45	No	N
4	9237-HQITU	Female	0	No	No	2	Yes	

5 rows × 21 columns

ENCODE CATEGORIES for ML

In [3]:
```
from sklearn.preprocessing import LabelEncoder

categorical_cols = ['gender','partner','dependents','phone_service','multiple_lines
    'internet_service','online_security','online_backup','device_protection',
    'tech_support','streaming_tv','streaming_movies','contract',
    'paperless_billing','payment_method','churn']

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

FEATURES and TARGET

In [4]:
```
x = df.drop(['customer_id', 'churn'], axis=1)
y = df['churn']
```

BINARY CLASSIFICATION MODEL for CHURN

Train/Test Split & Model Training

In [5]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_sta
model = RandomForestClassifier(random_state=42)
model.fit(x_train, y_train)
print(f"Train Accuracy: {model.score(x_train, y_train)}")
print(f"Test Accuracy: {model.score(x_test, y_test)}")
```

```
Train Accuracy: 0.9984025559105432
Test Accuracy: 0.7920511000709723
```

MODEL EXPLAINABILITY (ELI5/SHAP)

Using ELI5

In [6]:
```
import eli5
from eli5.sklearn import PermutationImportance

perm = PermutationImportance(model, random_state=42).fit(x_test, y_test)
eli5.show_weights(perm, feature_names=x.columns.tolist())
```

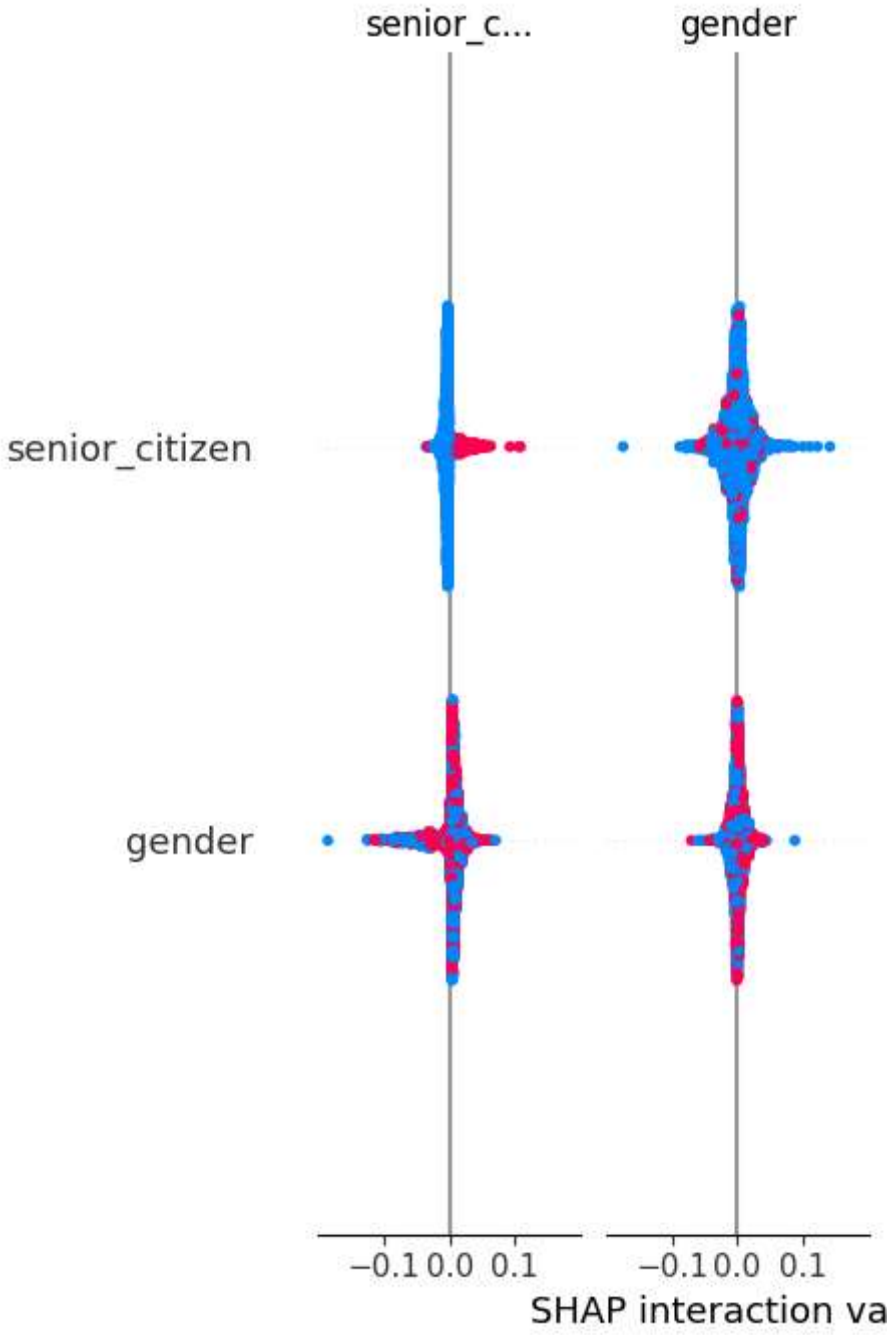
Out[6]:

Weight	Feature
0.0172 ± 0.0066	tenure
0.0163 ± 0.0187	contract
0.0075 ± 0.0105	online_security
0.0065 ± 0.0080	monthly_charges
0.0062 ± 0.0066	internet_service
0.0061 ± 0.0088	tech_support
0.0037 ± 0.0032	senior_citizen
0.0011 ± 0.0053	gender
0.0003 ± 0.0011	phone_service
-0.0004 ± 0.0050	partner
-0.0006 ± 0.0030	payment_method
-0.0016 ± 0.0071	paperless_billing
-0.0016 ± 0.0017	device_protection
-0.0023 ± 0.0053	streaming_tv
-0.0034 ± 0.0045	online_backup
-0.0045 ± 0.0023	multiple_lines
-0.0065 ± 0.0055	streaming_movies
-0.0068 ± 0.0130	total_charges
-0.0074 ± 0.0031	dependents

Using SHAP

In [7]:
```
import shap

explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(x_test)
shap.summary_plot(shap_values, x_test)
```



Customer Segmentation

In [8]:
```
# Default all as Dormant
df['segment'] = 'Dormant'
# At Risk criteria: low tenure & high monthly charges
df.loc[(df['tenure'] < 12) & (df['monthly_charges'] > df['monthly_charges'].mean())
# Loyal criteria: long tenure & not churned
df.loc[(df['tenure'] >= 24) & (df['churn'] == 0), 'segment'] = 'Loyal'

df['segment'].value_counts()
```

Out[8]:
```
segment
Loyal      3366
Dormant    2722
At Risk     955
Name: count, dtype: int64
```

Saving Charts

In [9]:
```
import matplotlib.pyplot as plt
plt.savefig('churn_rate_chart.png')
```

```
<Figure size 640x480 with 0 Axes>
```

Tables (Data Frames)

In [10]:
```
df.head().to_csv('sample_table.csv')
```

In [12]:
```
importances = model.feature_importances_
features = x.columns
feat_imp = pd.DataFrame({'Feature': features, 'Importance': importances})
feat_imp = feat_imp.sort_values('Importance', ascending=False)
feat_imp.to_csv('feature_importance.csv', index=False)
```