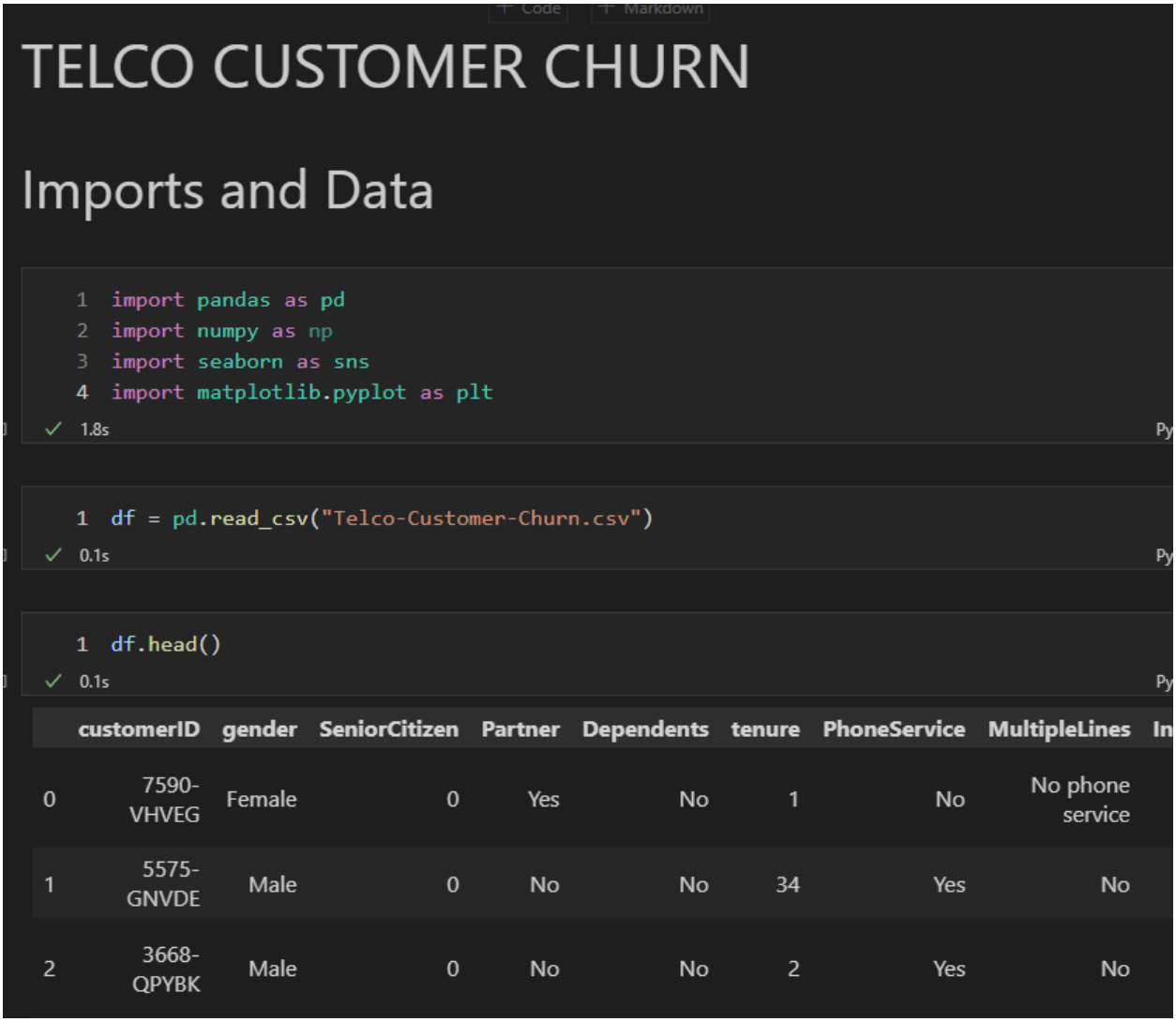
Overview



	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	In
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	

```
1  df.info()
```

✓ 0.1s

```
Output exceeds the size limit. Open the full output data
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7032 entries, 0 to 7031
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7032 non-null   object
 1   gender            7032 non-null   object
 2   SeniorCitizen     7032 non-null   int64
 3   Partner           7032 non-null   object
 4   Dependents        7032 non-null   object
 5   tenure            7032 non-null   int64
 6   PhoneService      7032 non-null   object
 7   MultipleLines     7032 non-null   object
 8   InternetService   7032 non-null   object
 9   OnlineSecurity    7032 non-null   object
...
 19  TotalCharges      7032 non-null   float64
 20  Churn             7032 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

```
1  df.isnull().sum().sum()
2  # Null değer yok
```

✓ 0.1s

```
0
```

```
1  df.isna().sum().sum()
2  # NaN değer yok
```

✓ 0.2s

```
0
```

```
1 df.describe()
```

✓ 0.1s

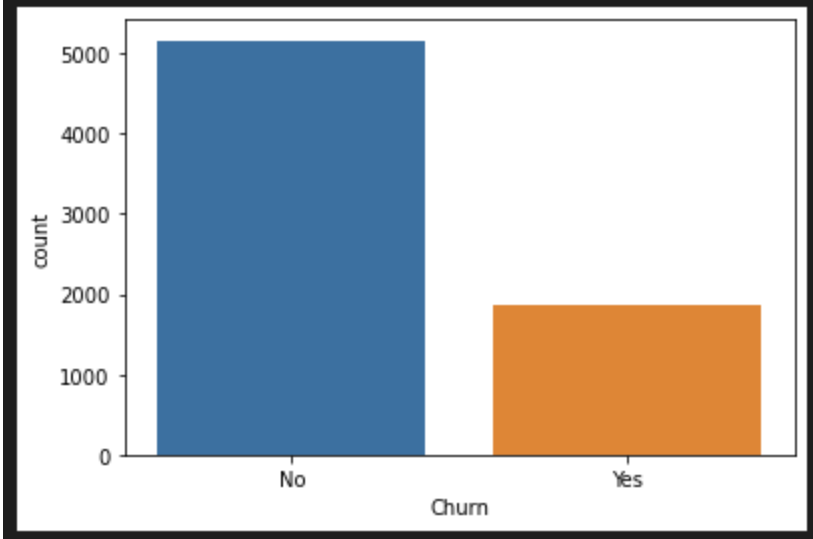
	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7032.000000	7032.000000	7032.000000	7032.000000
mean	0.162400	32.421786	64.798208	2283.300441
std	0.368844	24.545260	30.085974	2266.771362
min	0.000000	1.000000	18.250000	18.800000
25%	0.000000	9.000000	35.587500	401.450000
50%	0.000000	29.000000	70.350000	1397.475000
75%	0.000000	55.000000	89.862500	3794.737500
max	1.000000	72.000000	118.750000	8684.800000

EDA

```
1 sns.countplot(data=df, x="Churn")
```

✓ 0.3s

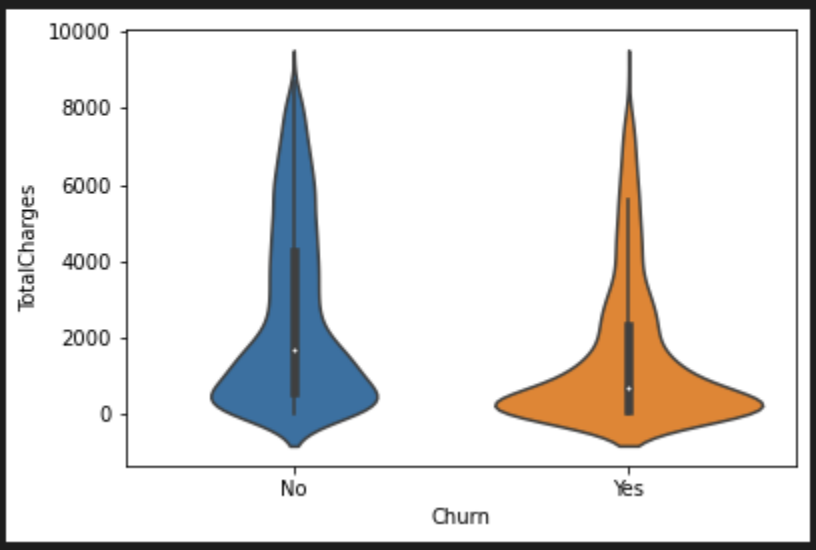
```
<AxesSubplot:xlabel='Churn', ylabel='count'>
```



```
1  sns.violinplot(data=df, x="Churn", y="TotalCharges")
```

✓ 0.4s

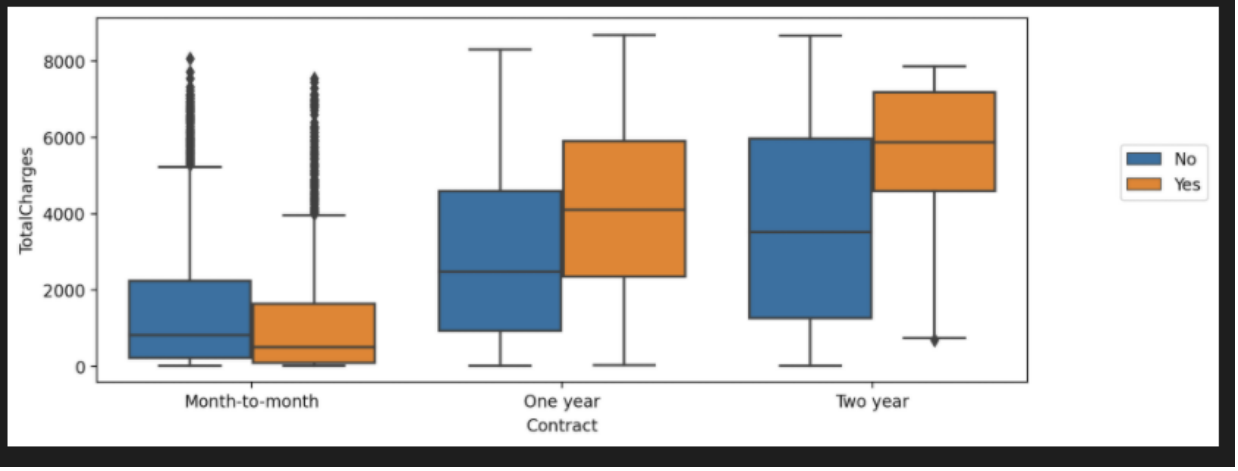
```
<AxesSubplot:xlabel='Churn', ylabel='TotalCharges'>
```




```
1  plt.figure(figsize=(10,4),dpi=200)
2  sns.boxplot(data=df, y="TotalCharges", x="Contract", hue="Churn")
3  plt.legend(loc=(1.1,0.5))
```

✓ 0.4s Python

```
<matplotlib.legend.Legend at 0x22ad2c24e50>
```



```
1  df.columns
```

✓ 0.7s

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

```
1  corr = pd.get_dummies(df[['gender', 'SeniorCitizen', 'Partner',
2   'Dependents','PhoneService', 'MultipleLines', 'InternetService',
3   'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
4   'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
5   'PaymentMethod','Churn']]).corr()
```

✓ 0.1s

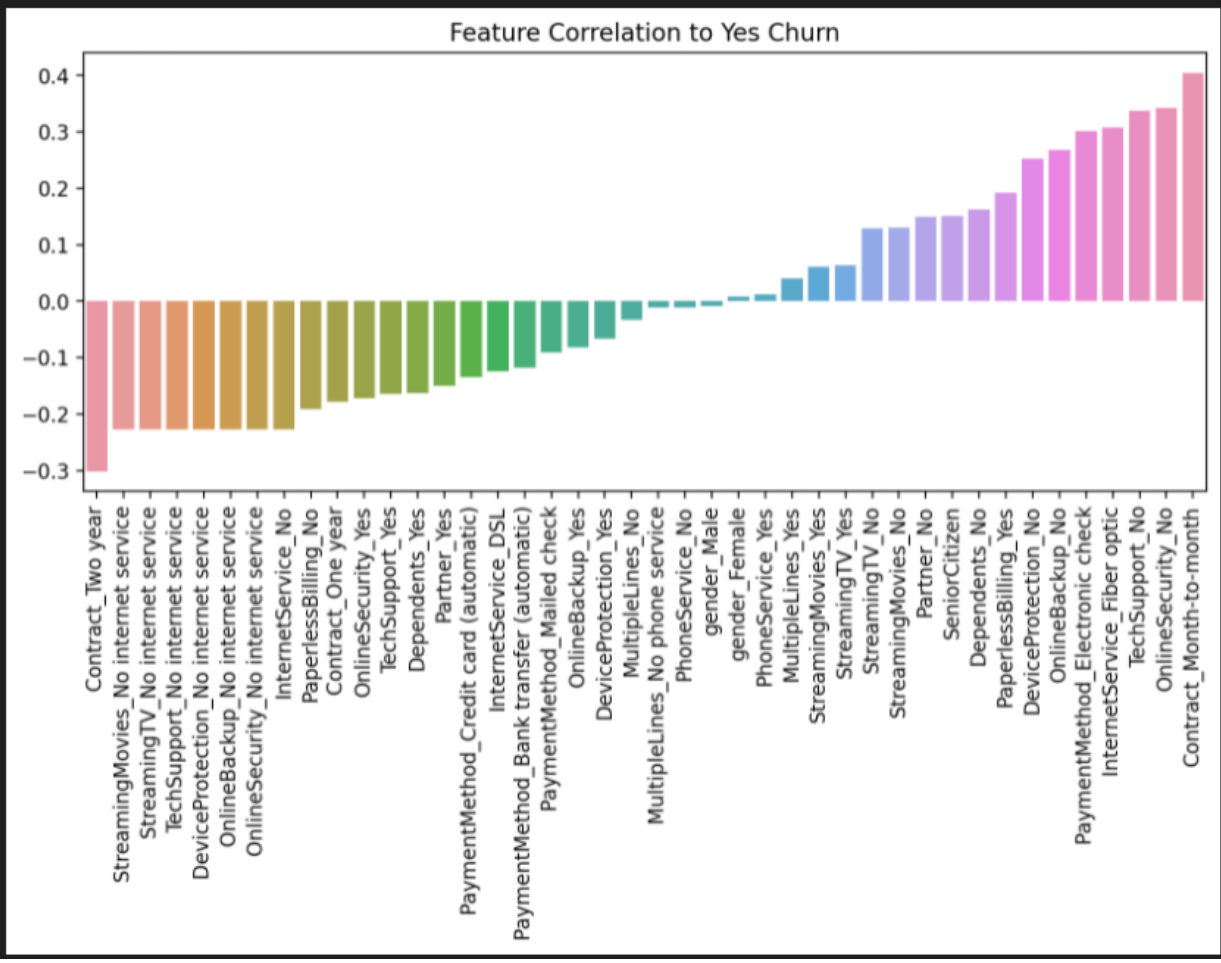
```python
corr["Churn_Yes"].sort_values().iloc[1:-1]
# Churn_Yes kendisi ile 1 tam korale Churn_No ile de1 tam korale
# ... oolduğu için 1. ve sonundu indeksteki veriler bizim için anlam
# ... ifade etmez. Bunları atmak için de iloc[1:-1] kullandık.
# ... yani 0. indeksi dışarıda bırakacak şekilde 1den -1. indekse kadar
# ... olan serinin tamamını aldık.
```

✓ 0.1s

```
Output exceeds the size limit. Open the full output data in a text editor
Contract_Two year                          -0.301552
StreamingMovies_No internet service        -0.227578
StreamingTV_No internet service            -0.227578
TechSupport_No internet service            -0.227578
DeviceProtection_No internet service       -0.227578
OnlineBackup_No internet service           -0.227578
OnlineSecurity_No internet service         -0.227578
InternetService_No                         -0.227578
PaperlessBilling_No                        -0.191454
Contract_One year                          -0.178225
OnlineSecurity_Yes                         -0.171270
TechSupport_Yes                            -0.164716
Dependents_Yes                             -0.163128
Partner_Yes                                -0.149982
PaymentMethod_Credit card (automatic)      -0.134687
...
InternetService_Fiber optic                 0.307463
TechSupport_No                              0.336877
OnlineSecurity_No                           0.342235
Contract_Month-to-month                     0.404565
Name: Churn_Yes, dtype: float64
```

```python
plt.figure(figsize=(10,4),dpi=200)
sns.barplot(x=corr["Churn_Yes"].sort_values().iloc[1:-1].index,
            y=corr['Churn_Yes'].sort_values().iloc[1:-1].values)
plt.title("Feature Correlation to Yes Churn")
plt.xticks(rotation=90);
```

✓ 1.5s Python



Churn Analysis

```
1 df["Contract"].unique()
```

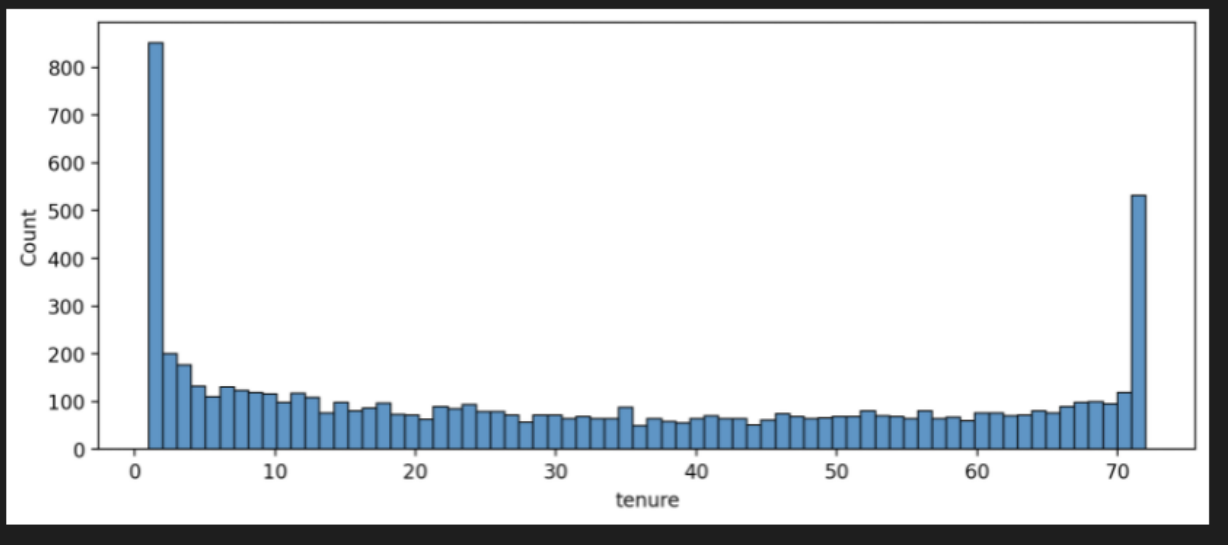
✓ 0.7s Python

```
array(['Month-to-month', 'One year', 'Two year'], dtype=object)
```

```
1 plt.figure(figsize=(10,4),dpi=200)
2 sns.histplot(data=df, x="tenure", bins=70)
```

✓ 0.6s Python

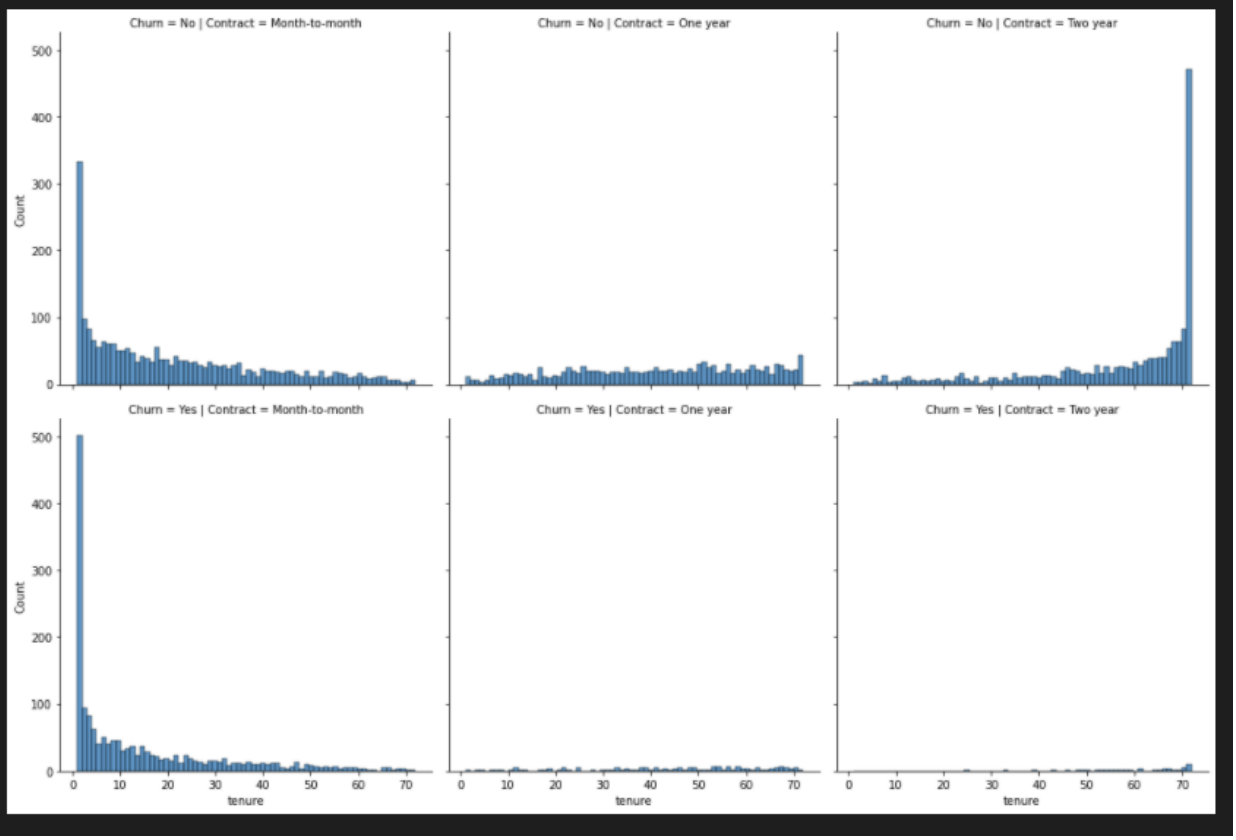
```
<AxesSubplot:xlabel='tenure', ylabel='Count'>
```



```
1  plt.figure(figsize=(10,4),dpi=200)
2  sns.displot(data=df, x="tenure",bins=70, col="Contract", row="Churn");
```

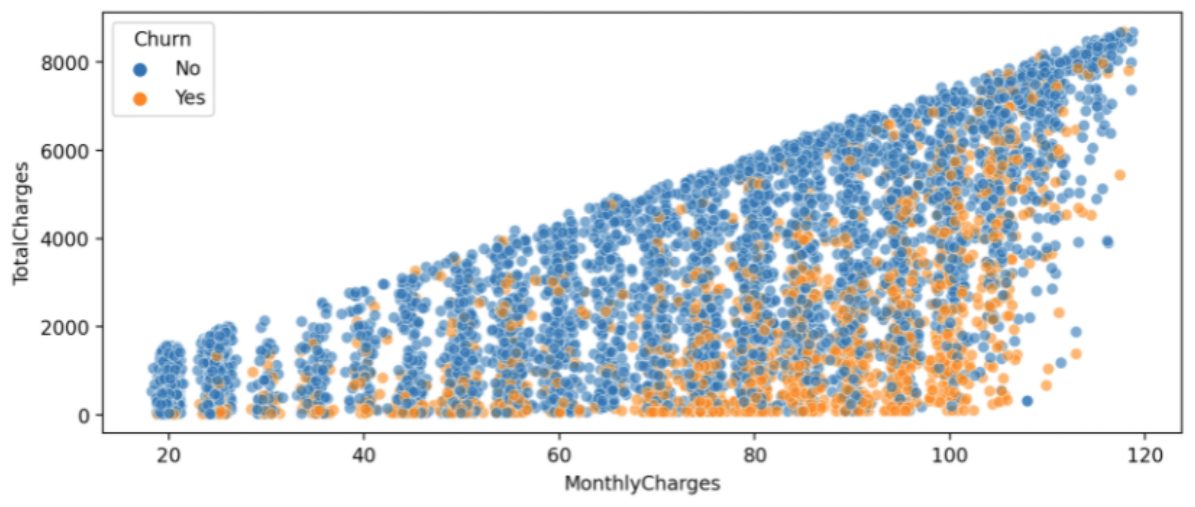
✓ 2.7s Python

```
<Figure size 2000x800 with 0 Axes>
```



```python
plt.figure(figsize=(10,4),dpi=200)
sns.scatterplot(data=df, x="MonthlyCharges", y="TotalCharges",
    hue="Churn", lw=0.5, alpha=0.6);
```

✓ 0.7s Python




```python
no_churn = df.groupby(["Churn","tenure"]).count().transpose()["No"]
yes_churn = df.groupby(["Churn","tenure"]).count().transpose()["Yes"]
```

✓ 0.1s Pyth

```python
yes_churn
```

✓ 0.1s Pyth

tenure	1	2	3	4	5	6	7	8	9	10	...	63	64	65	66	67	68	69	70	71
customerID	380	123	94	83	64	40	51	42	46	45	...	4	4	9	13	10	9	8	11	6
gender	380	123	94	83	64	40	51	42	46	45	...	4	4	9	13	10	9	8	11	6
SeniorCitizen	380	123	94	83	64	40	51	42	46	45	...	4	4	9	13	10	9	8	11	6
Partner	380	123	94	83	64	40	51	42	46	45	...	4	4	9	13	10	9	8	11	6
Dependents	380	123	94	83	64	40	51	42	46	45	...	4	4	9	13	10	9	8	11	6
PhoneService	380	123	94	83	64	40	51	42	46	45	...	4	4	9	13	10	9	8	11	6
MultipleLines	380	123	94	83	64	40	51	42	46	45	...	4	4	9	13	10	9	8	11	6
InternetService	380	123	94	83	64	40	51	42	46	45	...	4	4	9	13	10	9	8	11	6
OnlineSecurity	380	123	94	83	64	40	51	42	46	45	...	4	4	9	13	10	9	8	11	6
OnlineBackup	380	123	94	83	64	40	51	42	46	45	...	4	4	9	13	10	9	8	11	6

```
1  churn_rate = 100 * yes_churn / (yes_churn + no_churn)
```

✓ 0.9s

```
1  churn_rate.transpose()["customerID"]
```

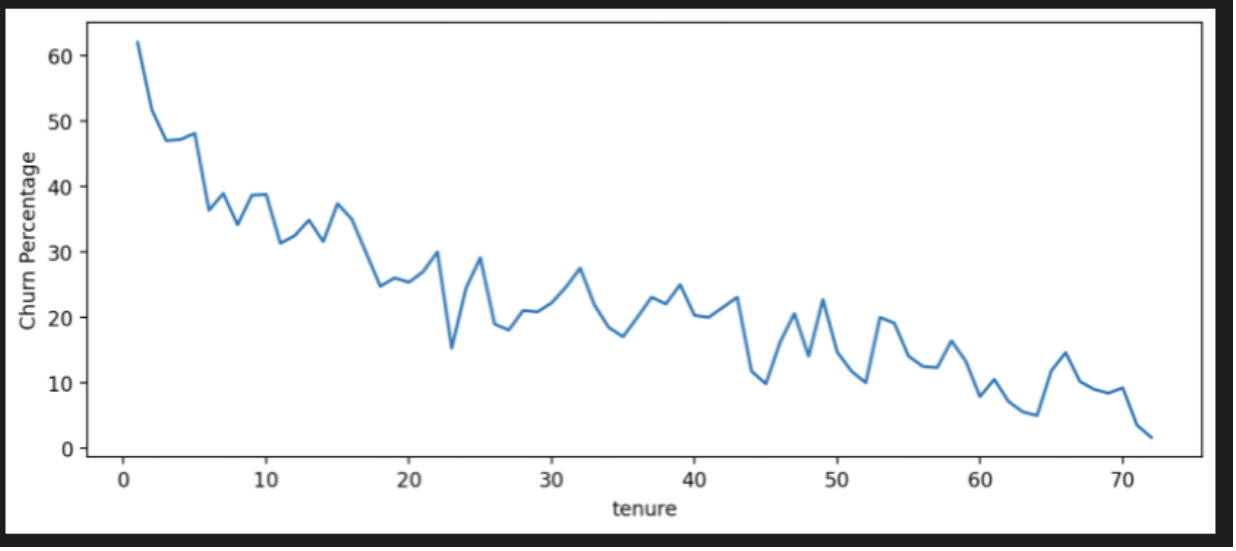
✓ 0.1s

```
tenure
1     61.990212
2     51.680672
3     47.000000
4     47.159091
5     48.120301
        ...
68     9.000000
69     8.421053
70     9.243697
71     3.529412
72     1.657459
Name: customerID, Length: 72, dtype: float64
```

```
1  plt.figure(figsize=(10,4),dpi=200)
2  churn_rate.iloc[0].plot()
3  plt.ylabel("Churn Percentage")
```

✓ 0.3s Python

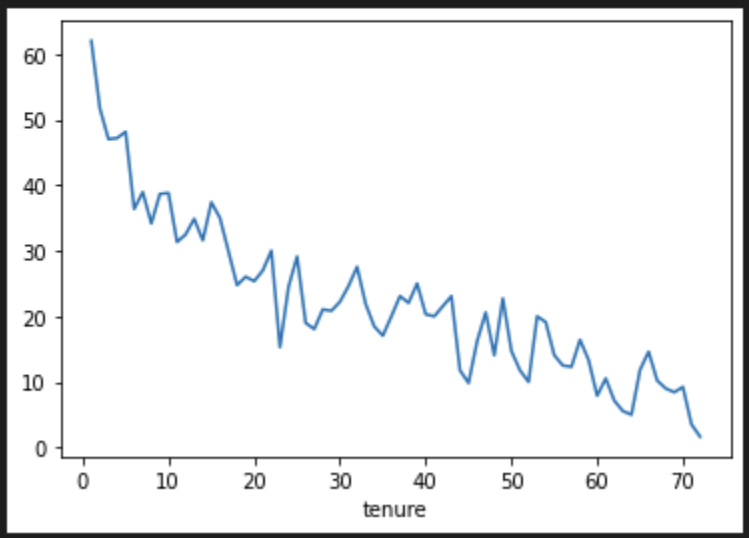
```
Text(0, 0.5, 'Churn Percentage')
```




```
1  churn_rate.transpose()["customerID"].plot()
```

✓ 0.3s

```
<AxesSubplot:xlabel='tenure'>
```



```python
def cohort(tenure):
    if tenure < 13:
        return "0-12 Months"
    elif tenure < 25:
        return '12-24 Months'
    elif tenure < 49:
        return '24-48 Months'
    else:
        return "Over 48 Months"
```

✓ 0.1s

```python
df['Tenure Cohort'] = df['tenure'].apply(cohort)
```

✓ 0.2s

```python
df.head(10)[['tenure','Tenure Cohort']]
```

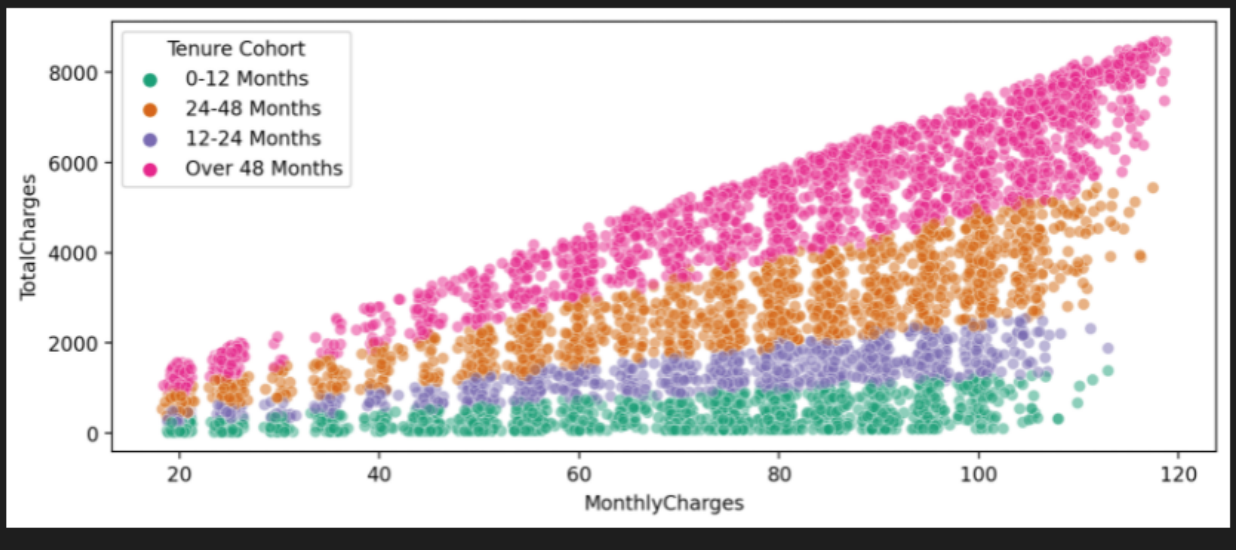
✓ 0.1s

	tenure	Tenure Cohort
0	1	0-12 Months
1	34	24-48 Months
2	2	0-12 Months
3	45	24-48 Months
4	2	0-12 Months
5	8	0-12 Months
6	22	12-24 Months
7	10	0-12 Months
8	28	24-48 Months
9	62	Over 48 Months

```
1 plt.figure(figsize=(10,4),dpi=200)
2 sns.scatterplot(data=df,x='MonthlyCharges',y='TotalCharges',
3     hue='Tenure Cohort', linewidth=0.5,alpha=0.5,palette='Dark2')
```

✓ 1.2s Pytho

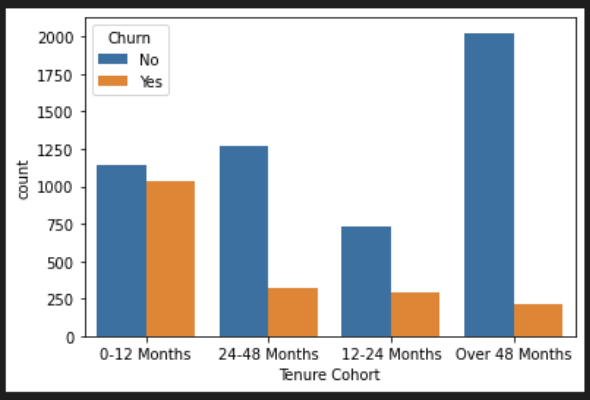
```
<AxesSubplot:xlabel='MonthlyCharges', ylabel='TotalCharges'>
```



```
1 sns.countplot(data=df, x="Tenure Cohort", hue="Churn")
```

✓ 0.7s

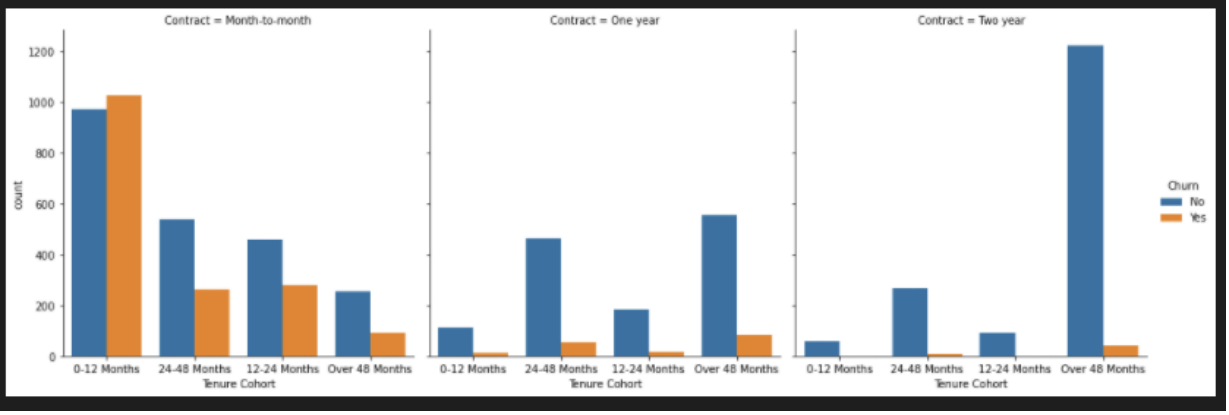
```
<AxesSubplot:xlabel='Tenure Cohort', ylabel='count'>
```




```
1 sns.catplot(data=df, x="Tenure Cohort", hue="Churn",
2     kind="count", col="Contract")
```

✓ 0.8s

```
<seaborn.axisgrid.FacetGrid at 0x22ad3a98d00>
```



Single Decision Tree

```python
X = df.drop(["Churn","customerID"], axis=1)
y = df["Churn"]
```

✓ 0.6s Python

```python
X = pd.get_dummies(X, drop_first=True)
```

✓ 0.1s Python

```python
from sklearn.model_selection import train_test_split
```

✓ 0.3s Python

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=101)
```

✓ 0.1s Python

```python
from sklearn.tree import DecisionTreeClassifier
```

✓ 0.1s Python

```python
dt = DecisionTreeClassifier(max_depth=6)
```

✓ 0.9s Python

```python
dt.fit(X_train, y_train)
```

✓ 0.2s Python

```
DecisionTreeClassifier(max_depth=6)
```

```python
preds = dt.predict(X_test)
```

✓ 0.1s Python

```
1 m sklearn.metrics import classification_report, plot_confusion_matrix, accuracy_score
```

✓ 0.7s

```
1 print(classification_report(y_test,preds))
```

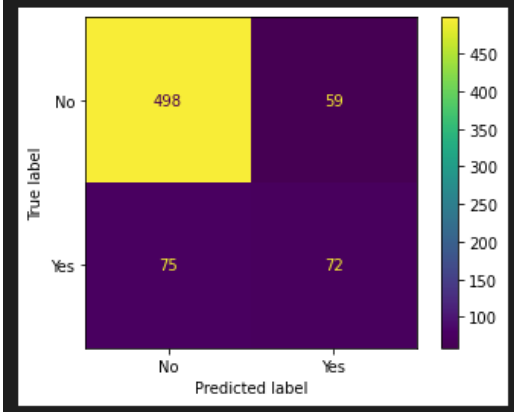
✓ 0.1s

```
              precision    recall  f1-score   support

          No       0.87      0.89      0.88       557
         Yes       0.55      0.49      0.52       147

    accuracy                           0.81       704
   macro avg       0.71      0.69      0.70       704
weighted avg       0.80      0.81      0.81       704
```

```
1 plot_confusion_matrix(dt,X_test,y_test)
```

✓ 0.8s

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x22ad2e4d160>
```

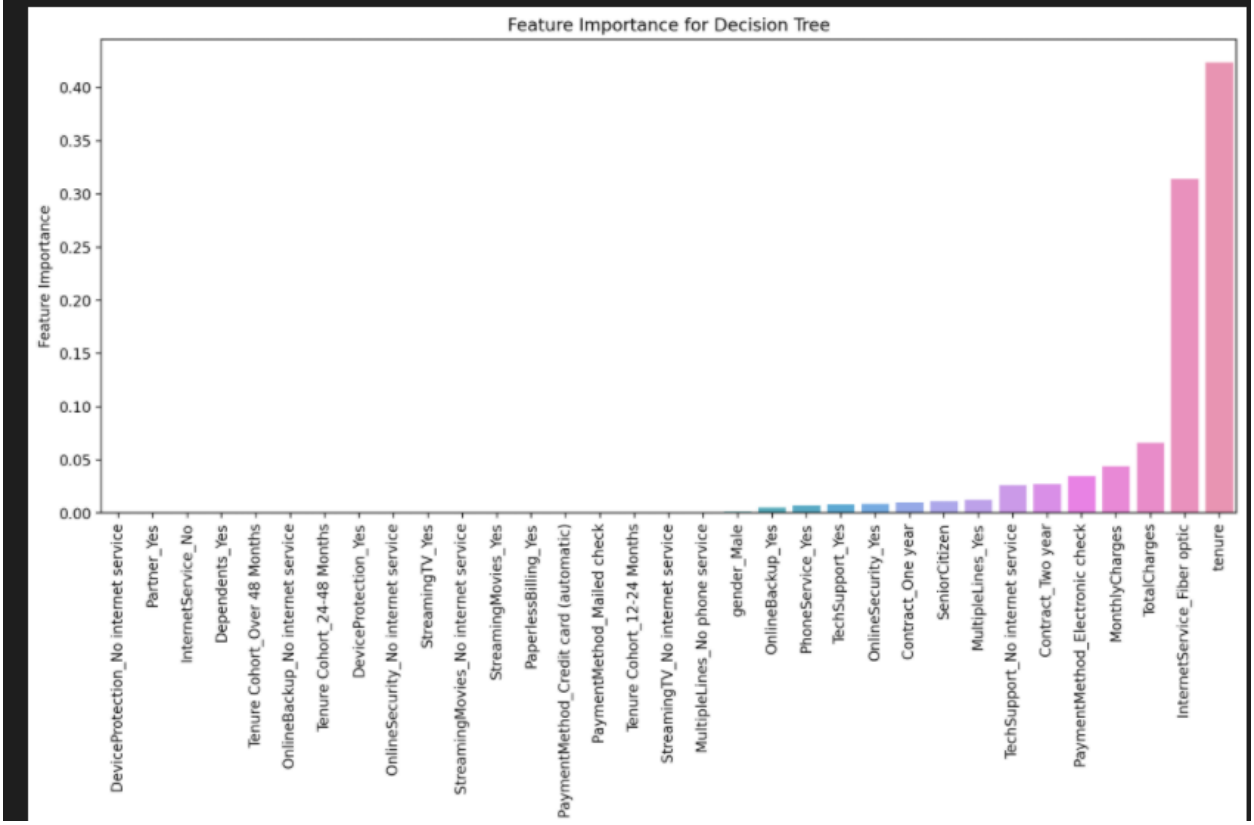


```
imp_feats = pd.DataFrame(data=dt.feature_importances_, index=X.columns,
    columns=["Feature Importance"]).sort_values("Feature Importance")
```

0.1s

```
plt.figure(figsize=(14,6),dpi=200)
sns.barplot(data=imp_feats.sort_values('Feature Importance'),
    x=imp_feats.sort_values('Feature Importance').index,y='Feature Importance')
plt.xticks(rotation=90)
plt.title("Feature Importance for Decision Tree");
```

1.1s

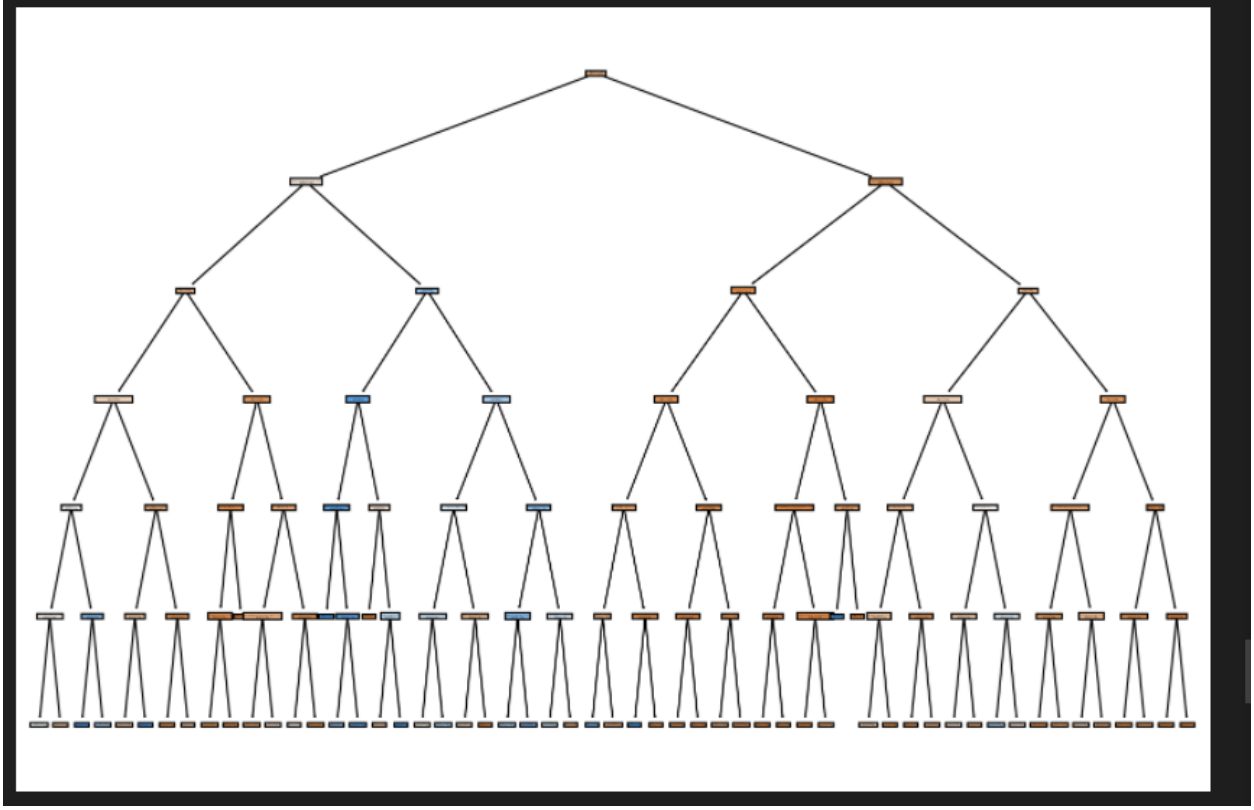


```python
from sklearn.tree import plot_tree
```

0.2s Python

```python
plt.figure(figsize=(12,8),dpi=150)
plot_tree(dt,filled=True, feature_names=X.columns);
```

9.4s Python



Random Forest

```
1  from sklearn.ensemble import RandomForestClassifier
```

✓ 0.9s

```
1  rf = RandomForestClassifier(n_estimators=100)
```

✓ 0.1s

```
1  rf.fit(X_train, y_train)
```

✓ 1.3s

```
RandomForestClassifier()
```

```
1  preds = rf.predict(X_test)
```

✓ 0.1s

```
1  print(classification_report(y_test,preds))
```

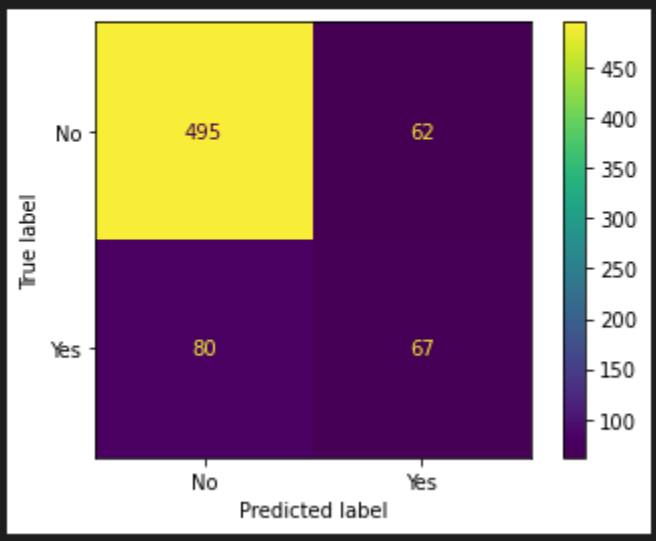
✓ 0.1s

```
              precision    recall  f1-score   support

          No       0.86      0.89      0.87       557
         Yes       0.52      0.46      0.49       147

    accuracy                           0.80       704
   macro avg       0.69      0.67      0.68       704
weighted avg       0.79      0.80      0.79       704
```

```
1  plot_confusion_matrix(rf,X_test,y_test)
```

✓ 0.6s

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatri
```



Boosted Trees

```
1 from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier
```
✓ 0.9s

```
1 ada_model = AdaBoostClassifier()
```
✓ 0.1s

```
1 ada_model.fit(X_train, y_train)
```
✓ 1.3s

```
AdaBoostClassifier()
```

```
1 preds = ada_model.predict(X_test)
```
✓ 0.1s

```
1 print(classification_report(y_test, preds))
```
✓ 0.2s

```
              precision    recall  f1-score   support

          No       0.88      0.90      0.89       557
         Yes       0.60      0.54      0.57       147

    accuracy                           0.83       704
   macro avg       0.74      0.72      0.73       704
weighted avg       0.82      0.83      0.83       704
```

```
1  plot_confusion_matrix(rf, X_test, y_test)
```

✓ 0.6s

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMa
```