↔

Support Vector Machines

▼ imports

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import seaborn as sns
```

✓ 16.7s

```
1  df = pd.read_csv("mouse_viral_study.csv")
```

✓ 0.1s

Data & Viz

```
1  df.head()
```

✓ 0.1s

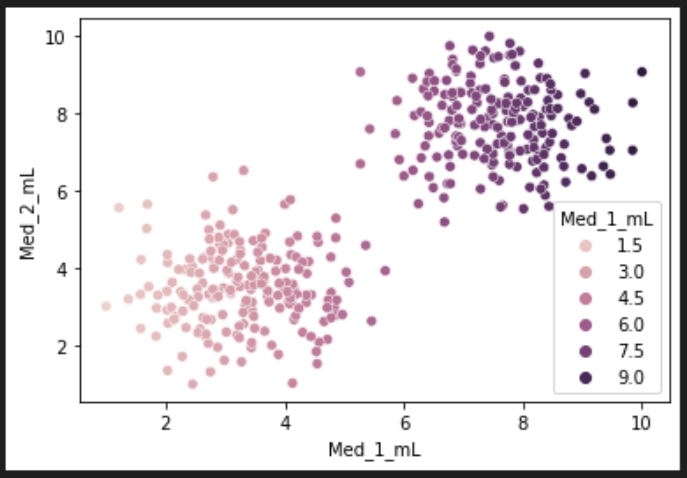
	Med_1_mL	Med_2_mL	Virus Present
0	6.508231	8.582531	0
1	4.126116	3.073459	1
2	6.427870	6.369758	0
3	3.672953	4.905215	1
4	1.580321	2.440562	1

▼ Data and Viz

```
1  sns.scatterplot(data=df, x="Med_1_mL", y="Med_2_mL", hue="Med_1_mL")
```

✓ 0.4s

```
<AxesSubplot:xlabel='Med_1_mL', ylabel='Med_2_mL'>
```



▼ SVM Model & Plot

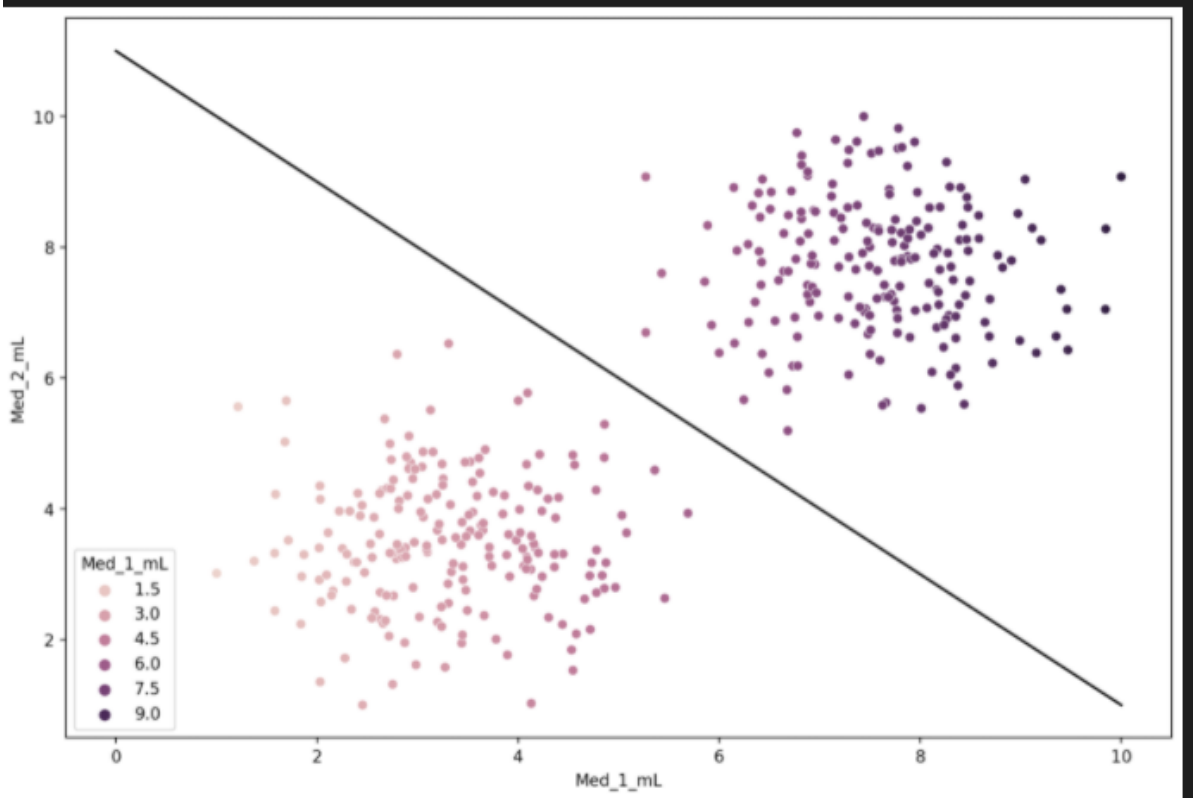
```python
plt.figure(figsize=(12,8), dpi=200)
sns.scatterplot(data=df, x="Med_1_mL", y="Med_2_mL", hue="Med_1_mL")

x = np.linspace(0,10,100)
m = -1
b = 11
y = m*x + b

plt.plot(x,y,"black")
```

✓ 0.7s Pytho

```
[<matplotlib.lines.Line2D at 0x1ef6d9e4b80>]
```



SVM

```
1  from sklearn.svm import SVC
```

✓ 0.3s

```
1  X = df.drop("Virus Present", axis=1)
2  y = df["Virus Present"]
```

✓ 0.3s

```
1  model = SVC(kernel="linear", C=1000)
```

✓ 0.5s

```
1  model.fit(X,y)
```

✓ 0.8s

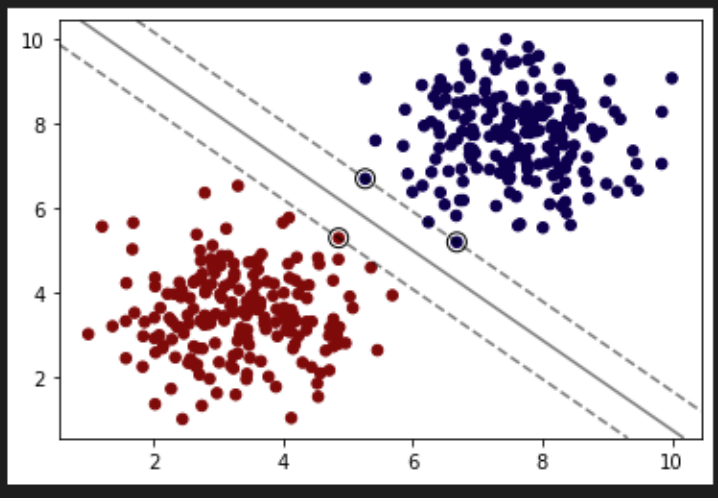
```
SVC(C=1000, kernel='linear')
```

```
1  from svm_margin_plot import plot_svm_boundary
```

✓ 0.1s

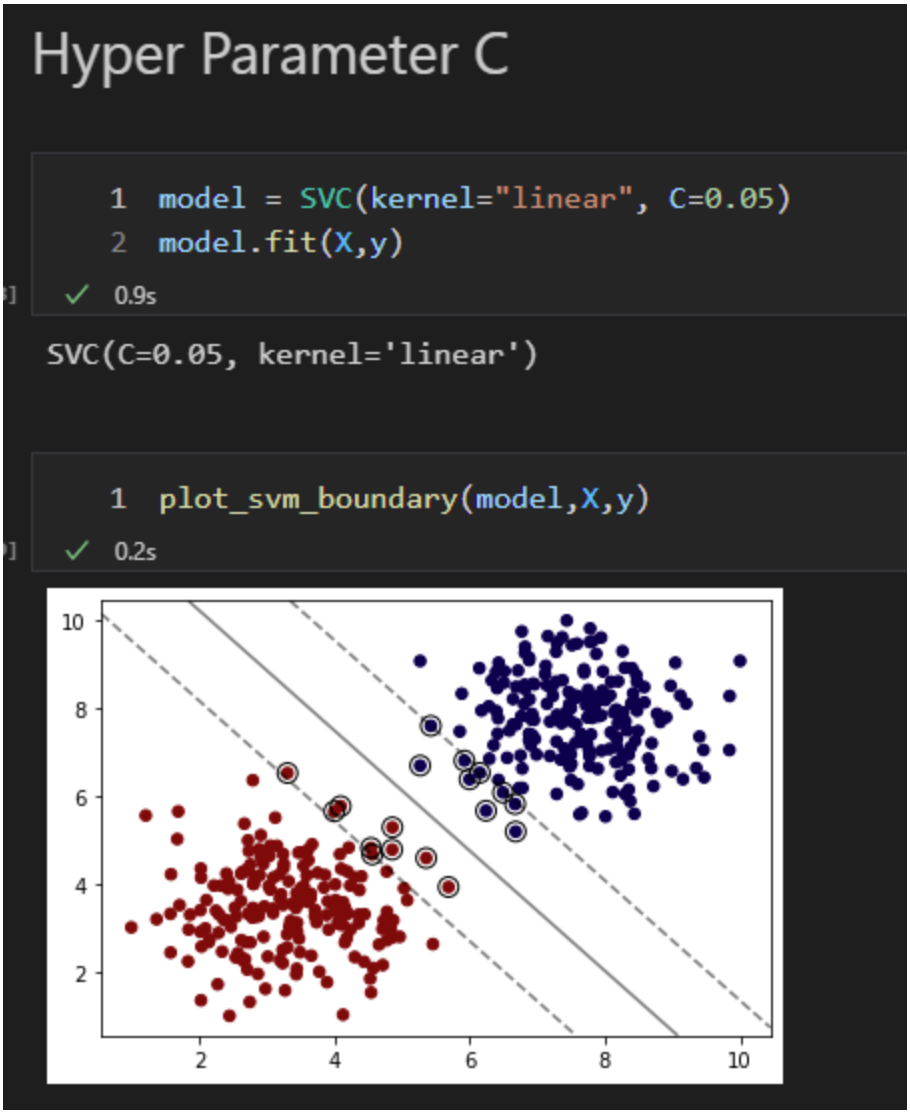
```
1  plot_svm_boundary(model,X,y)
```

✓ 0.3s



▼ Hyper parameter C

- C azaldıkça marjinler artar ve daha çok veriyi içine alır.

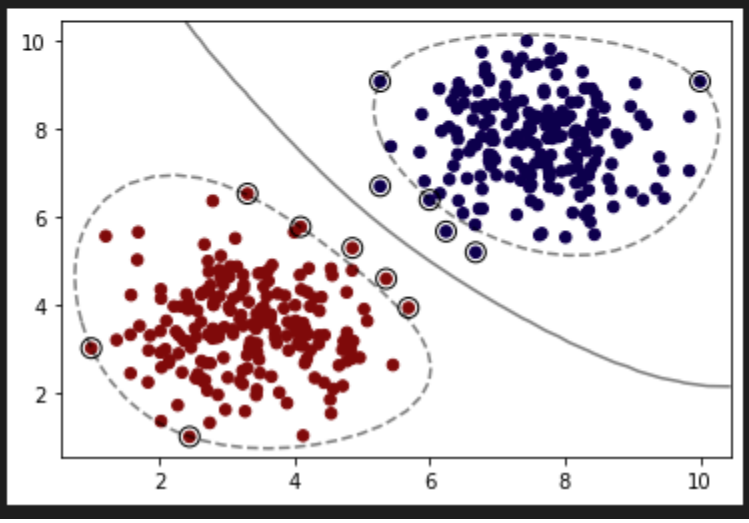


- “rbf” en ideal yöndem. (Default olan da bu)

```
1  model = SVC(kernel="rbf", C=1)
2  model.fit(X,y)
```
✓ 0.6s

```
SVC(C=1)
```

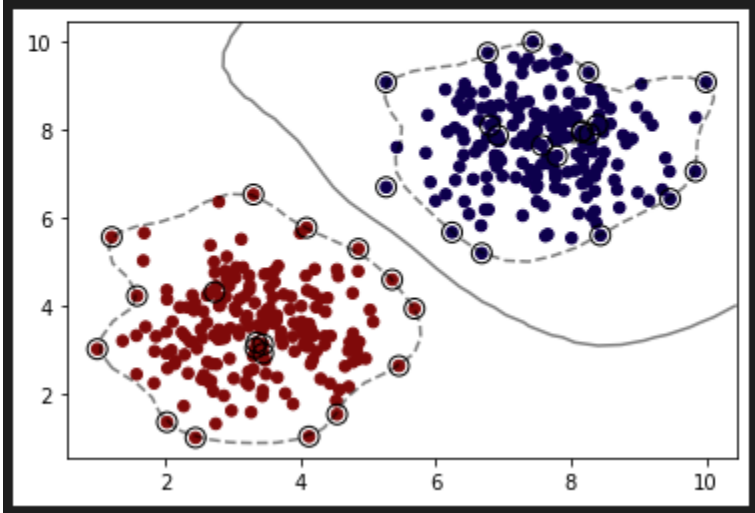
```
1  plot_svm_boundary(model,X,y)
```
✓ 0.4s



- Farklı gamma değerleri için sonuçlar (gamma büyüdükçe overfitting olur)

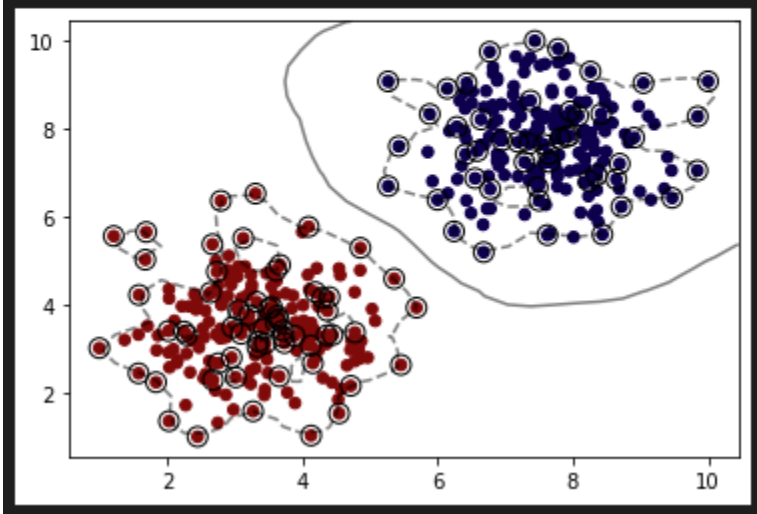
```
1  model = SVC(kernel="rbf", C=1, gamma=0.5)
2  model.fit(X,y)
3  plot_svm_boundary(model,X,y)
```

✓ 0.1s




```
1  model = SVC(kernel="rbf", C=1, gamma=1.5)
2  model.fit(X,y)
3  plot_svm_boundary(model,X,y)
```

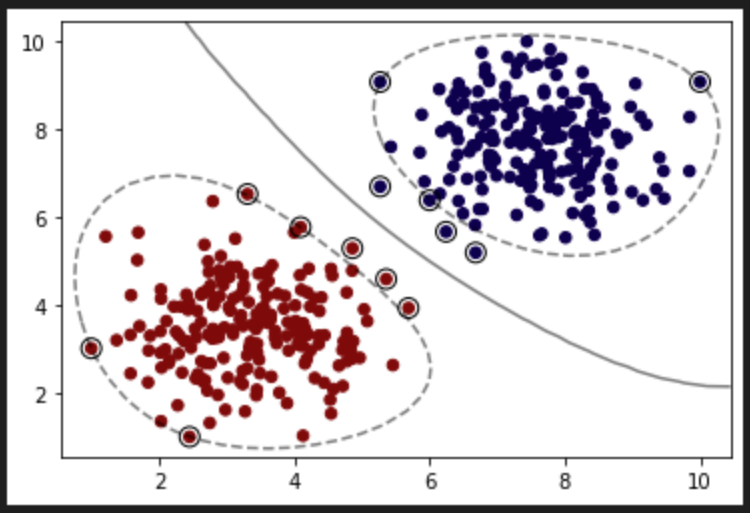
✓ 0.2s



- ideal olan gamma=”scale
- önce bununla başla

```
model = SVC(kernel="rbf", C=1, gamma="scale")
model.fit(X,y)
plot_svm_boundary(model,X,y)
```

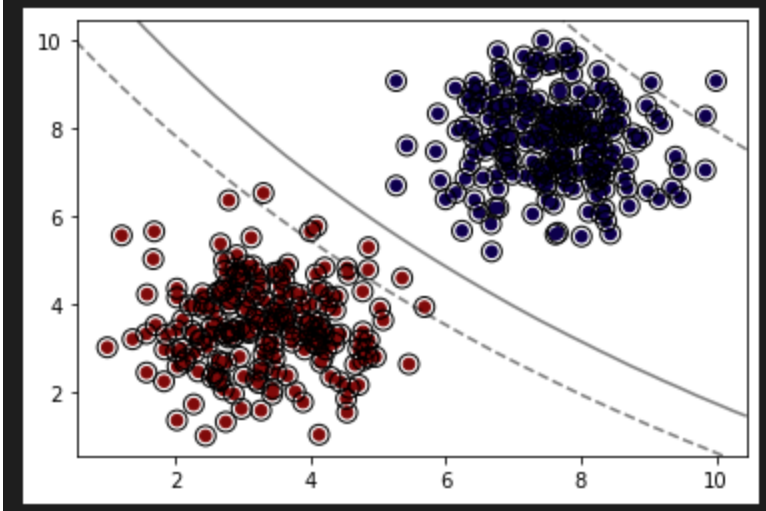
✓ 0.2s



- sigmoid, bütün veriler işaretlendi

```
model = SVC(kernel="sigmoid")
model.fit(X,y)
plot_svm_boundary(model,X,y)
```

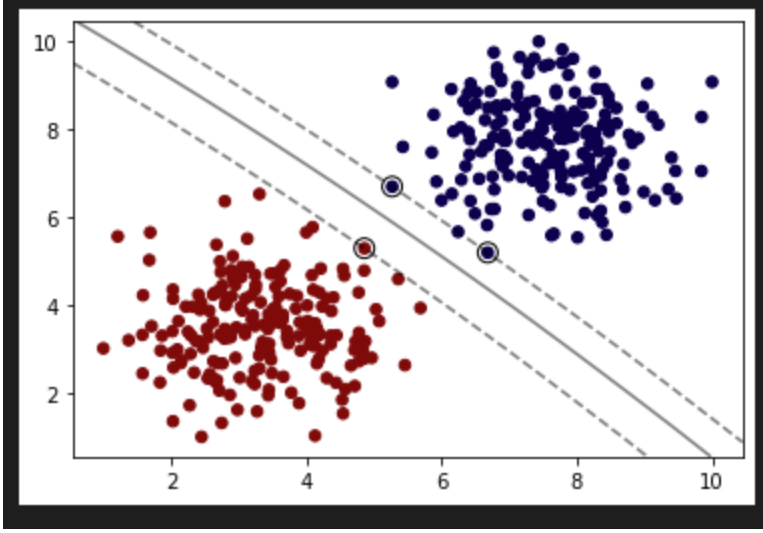
✓ 0.1s



- poly model

```
1  model = SVC(kernel="poly", C=0.05, degree=4)
2  model.fit(X,y)
3  plot_svm_boundary(model, X, y)
```

✓ 0.5s



▼ Grid CV

- Grid CV, Fit and best parameters

```
1 from sklearn.model_selection import GridSearchCV
```
✓ 0.2s

```
1 svm = SVC()
2 param_grid={
3     "C":[0.01,0.1,1],
4     "kernel":["linear","rbf"]
5 }
```
✓ 0.4s

```
1 grid = GridSearchCV(svm, param_grid)
```
✓ 0.3s

```
1 grid.fit(X,y)
```
✓ 0.3s

```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.01, 0.1, 1], 'kernel': ['linear', 'rbf']})
```

```
1 grid.best_params_
```
✓ 0.4s

```
{'C': 0.01, 'kernel': 'linear'}
```

▼ Next Page

-
-
-
-
-
-
-
-

-

-

-

-

-

▼ Cement Slump

Imports

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

✓ 1.8s Python

```python
df = pd.read_csv('cement_slump.csv')
```

✓ 0.1s Python

Data and Viz

```python
df.head()
```

✓ 0.7s Python

Cement	Slag	Fly ash	Water	SP	Coarse Aggr.	Fine Aggr.	SLUMP(cm)	FLOW(cm)	Compressive Strength (28-day) (Mpa)
273.0	82.0	105.0	210.0	9.0	904.0	680.0	23.0	62.0	34.99
163.0	149.0	191.0	180.0	12.0	843.0	746.0	0.0	20.0	41.14
162.0	148.0	191.0	179.0	16.0	840.0	743.0	1.0	20.0	41.81
162.0	148.0	190.0	179.0	19.0	838.0	741.0	3.0	21.5	42.08
154.0	112.0	144.0	220.0	10.0	923.0	658.0	20.0	64.0	26.82

```
1  df.corr()["Compressive Strength (28-day)(Mpa)"]
```

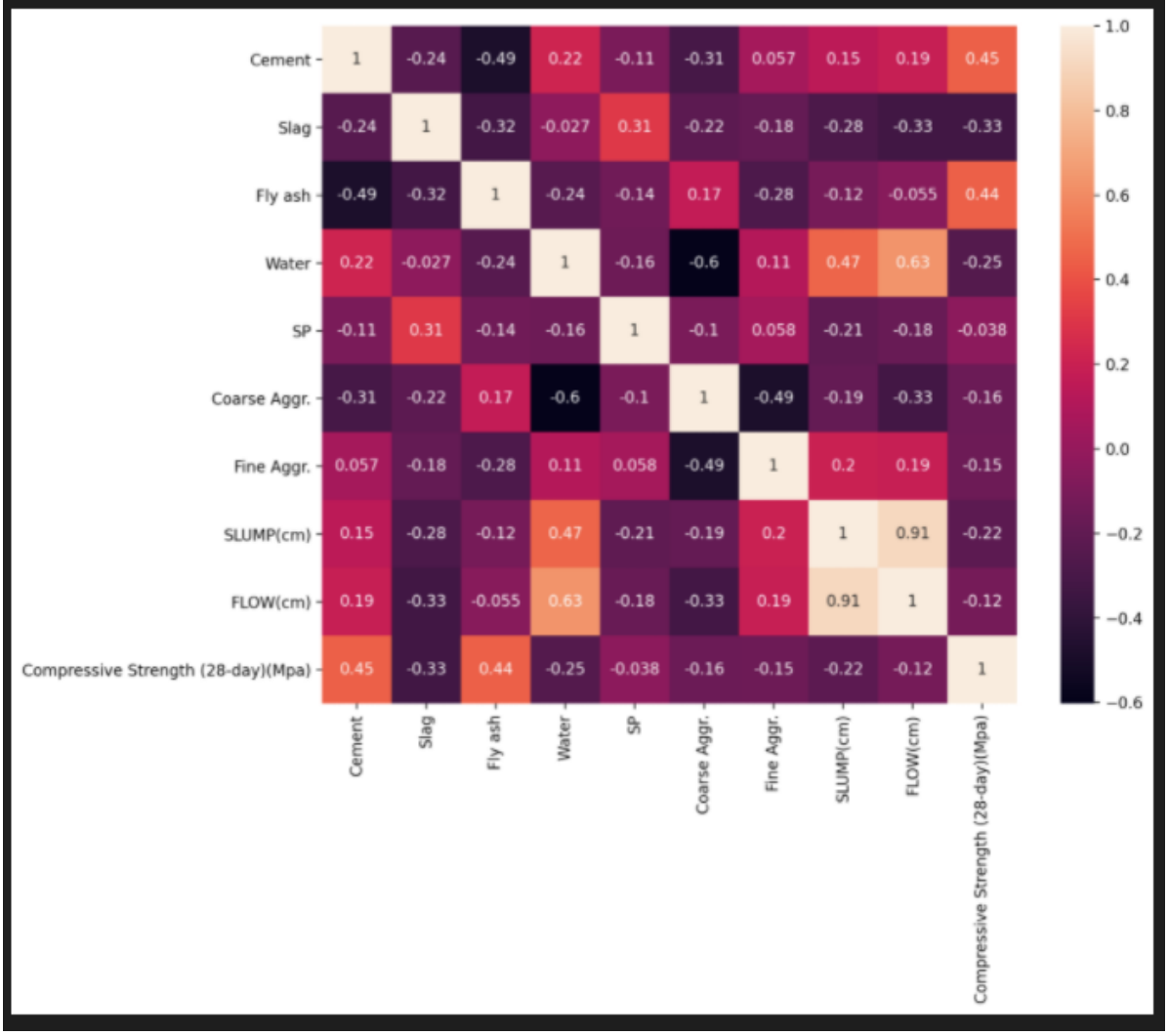
✓ 0.2s

```
Cement                                 0.445656
Slag                                  -0.331522
Fly ash                                0.444380
Water                                 -0.254320
SP                                    -0.037909
Coarse Aggr.                          -0.160610
Fine Aggr.                            -0.154532
SLUMP(cm)                             -0.223499
FLOW(cm)                              -0.124189
Compressive Strength (28-day)(Mpa)     1.000000
Name: Compressive Strength (28-day)(Mpa), dtype: float64
```

```
1  plt.figure(figsize=(10,8), dpi=200)
2  sns.heatmap(df.corr(), annot=True)
```

✓ 2.3s

```
<AxesSubplot:>
```



```
1  df.columns
```
✓ 0.8s

```
Index(['Cement', 'Slag', 'Fly ash', 'Water', 'SP', 'Coarse Aggr.',
       'Fine Aggr.', 'SLUMP(cm)', 'FLOW(cm)',
       'Compressive Strength (28-day)(Mpa)'],
      dtype='object')
```

Train Test Split

```
1  X = df.drop('Compressive Strength (28-day)(Mpa)',axis = 1)
2  y = df['Compressive Strength (28-day)(Mpa)']
```
✓ 0.9s

```
1  from sklearn.model_selection import train_test_split
```
✓ 0.3s

```
1  X_train, X_test, y_train, y_test = train_test_split(X, y,
2  | test_size=0.3, random_state=101)
```
✓ 0.4s

```
1  from sklearn.preprocessing import StandardScaler
```
✓ 0.3s

```
1  scaler = StandardScaler()
```
✓ 0.1s

```
1  scaled_X_train = scaler.fit_transform(X_train)
2  scaled_X_test = scaler.transform(X_test)
```
✓ 0.1s

```
1  from sklearn.svm import SVR, LinearSVR
2  # There are three different implementations of Support Vector
3  #... Regression: SVR, NuSVR and LinearSVR. LinearSVR provides a
4  #... faster implementation than SVR but only considers the linear kernel,
5  #... while NuSVR implements a slightly different formulation than SVR and LinearSVR.
```

```
1  base_model = SVR()
```
✓ 0.5s

```
1  base_model.fit(scaled_X_train,y_train)
```
✓ 0.1s

```
SVR()
```

```
1  base_preds = base_model.predict(scaled_X_test)
```
✓ 0.8s

Evaluation

```
1  from sklearn.metrics import mean_absolute_error, mean_squared_error
```
✓ 0.8s

+ Code + Markdown

```
1  mean_absolute_error(y_test,base_preds)
```
✓ 0.7s

```
5.236902091259179
```

```
1  np.sqrt(mean_squared_error(y_test,base_preds))
```
✓ 0.1s

```
6.695914838327133
```

```
1  y_test.mean()
```
✓ 0.8s

```
36.26870967741935
```

Better Grid Model

```
1  param_grid = {
2      'C':[0.001,0.01,0.1,0.5,1],
3      'kernel':['linear','rbf','poly'],
4      'gamma':['scale','auto'],
5      'degree':[2,3,4],
6      'epsilon':[0,0.01,0.1,0.5,1,2]
7  }
```

✓ 0.1s Python

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

✓ 0.2s Python

```
1  svr = SVR()
2  grid = GridSearchCV(svr,param_grid=param_grid)
```

✓ 0.1s Python

```
1  grid.fit(scaled_X_train, y_train)
```

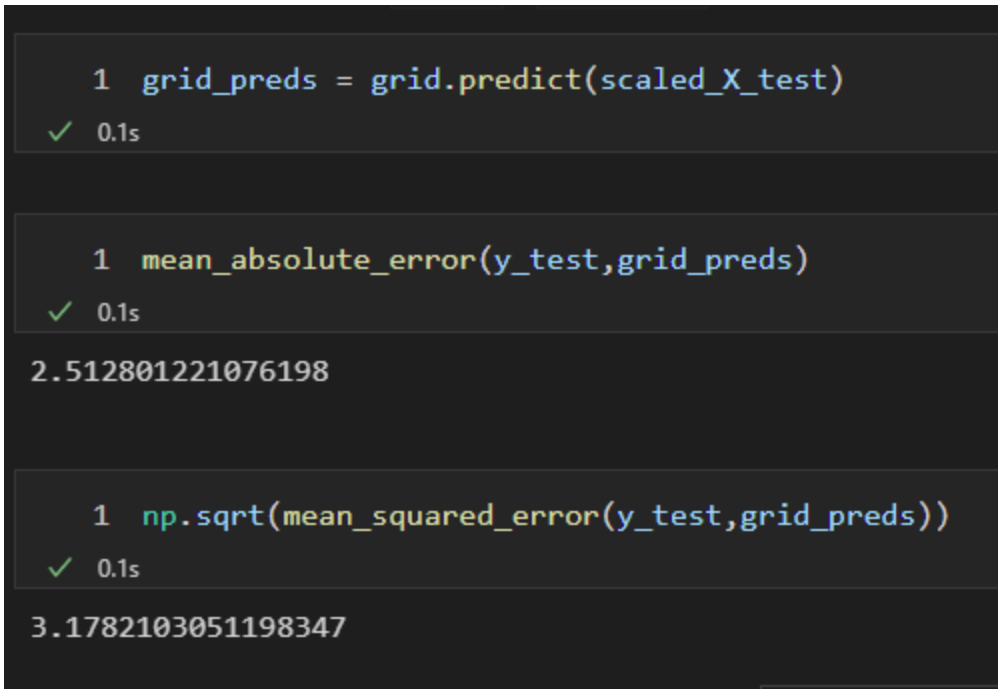
✓ 7.6s Python

```
GridSearchCV(estimator=SVR(),
             param_grid={'C': [0.001, 0.01, 0.1, 0.5, 1],
'degree': [2, 3, 4],
                         'epsilon': [0, 0.01, 0.1, 0.5, 1, 2],
                         'gamma': ['scale', 'auto'],
                         'kernel': ['linear', 'rbf', 'poly']})
```

```
1  grid.best_params_
```

✓ 0.1s Python

```
{'C': 1, 'degree': 2, 'epsilon': 2, 'gamma': 'scale',
'kernel': 'linear'}
```



▼ Next Page

-
-
-
-
-
-
-
-
-
-
-
-
-

▼ Wine Fraud Project

Imports

```
1  import numpy as np
2  import pandas as pd
3  import seaborn as sns
4  import matplotlib.pyplot as plt
```

✓ 11.3s

```
1  df = pd.read_csv("wine_fraud.csv")
```

✓ 0.2s

Data and Viz

```
1 df.head()
```

✓ 0.9s

ric id	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	Legit
00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	Legit
04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	Legit
56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	Legit
00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	Legit

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

✓ 0.6s

```
array(['Legit', 'Fraud'], dtype=object)
```

```
1 df["quality"].value_counts()
```

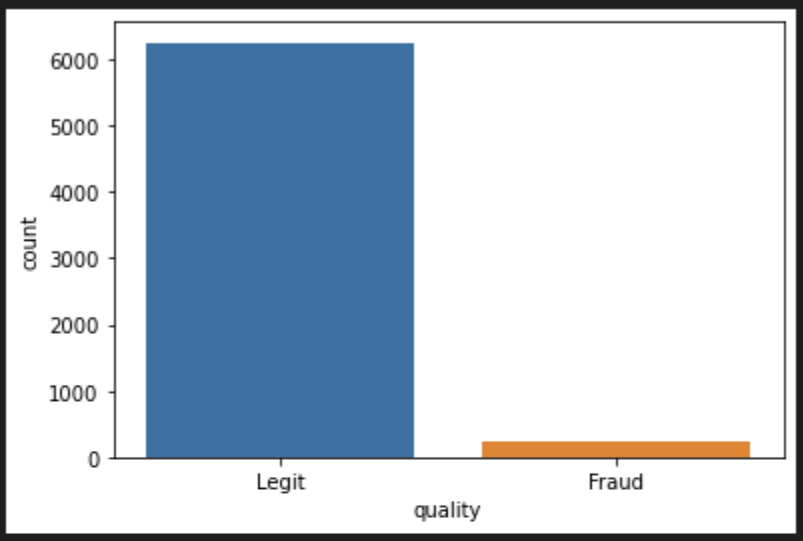
✓ 0.8s

```
Legit    6251
Fraud     246
Name: quality, dtype: int64
```

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

✓ 0.3s

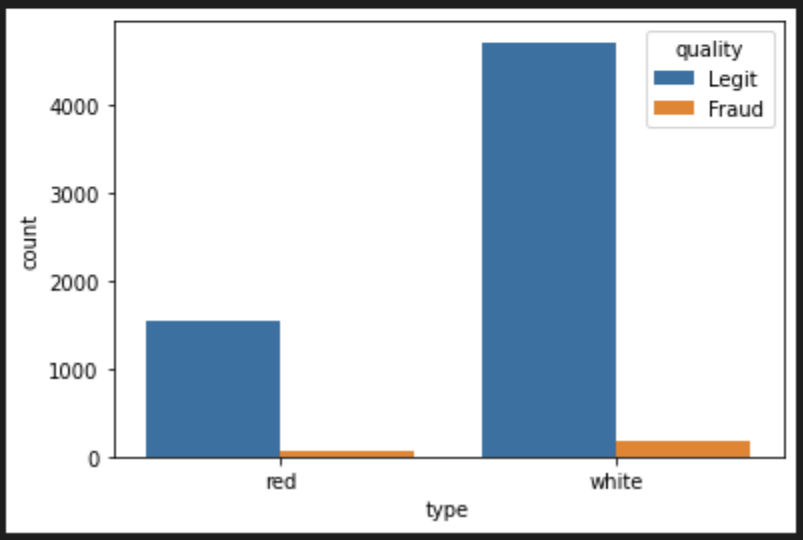
```
<AxesSubplot:xlabel='quality', ylabel='count'>
```




```
1  sns.countplot(x="type",hue="quality", data=df)
```

✓ 0.2s

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



```
1  reds = df[df["type"]=="red"]
2  whites = df[df["type"]=="white"]
```
✓ 0.3s

```
1  red_fraud = round(100*(len(reds[reds["quality"]=="Fraud"])/len(reds)),2)
2  white_fraud = round(100*(len(whites[whites["quality"]=="Fraud"])/len(whites)),2)
```
✓ 0.5s

```
1  print(f"{red_fraud}% of the red wines are fraud")
2  print("\n")
3  print(f"{white_fraud}% of the white wines are fraud")
```
✓ 0.1s

```
3.94% of the red wines are fraud


3.74% of the white wines are fraud
```

```
1  df["Fraud"]=df["quality"].map({"Legit":0, "Fraud":1})
```
✓ 0.7s

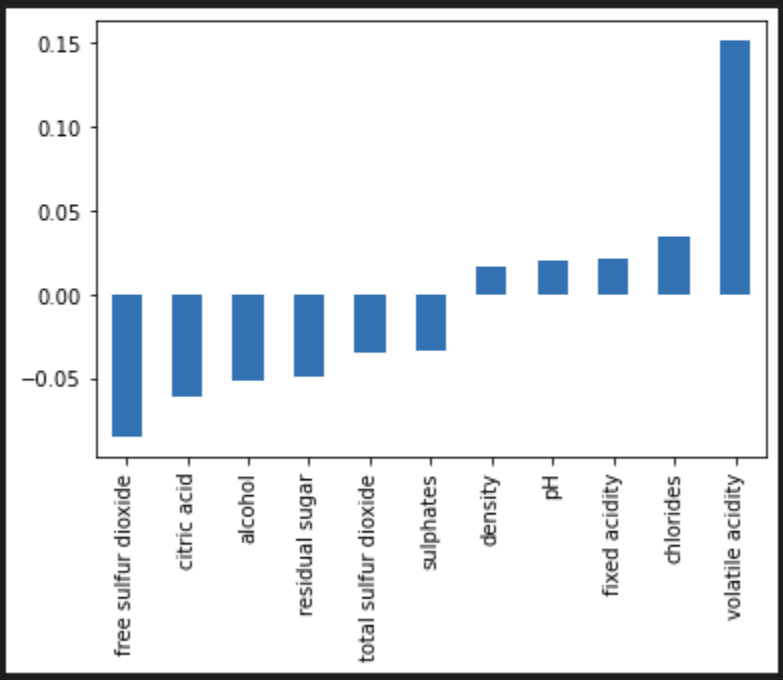
```
1  df.corr()['Fraud']
```
✓ 0.7s

```
fixed acidity           0.021794
volatile acidity        0.151228
citric acid            -0.061789
residual sugar         -0.048756
chlorides               0.034499
free sulfur dioxide    -0.085204
total sulfur dioxide   -0.035252
density                 0.016351
pH                      0.020107
sulphates              -0.034046
alcohol                -0.051141
Fraud                   1.000000
Name: Fraud, dtype: float64
```

```
1  df.corr()["Fraud"][:-1].sort_values().plot(kind="bar")
2  # [:-1] : sonuncu değeri atıyor fraud x fraud korelasyonu
3  # .... zaten 1 oldğu için gradikte işimize yaramaz
```

✓ 0.2s

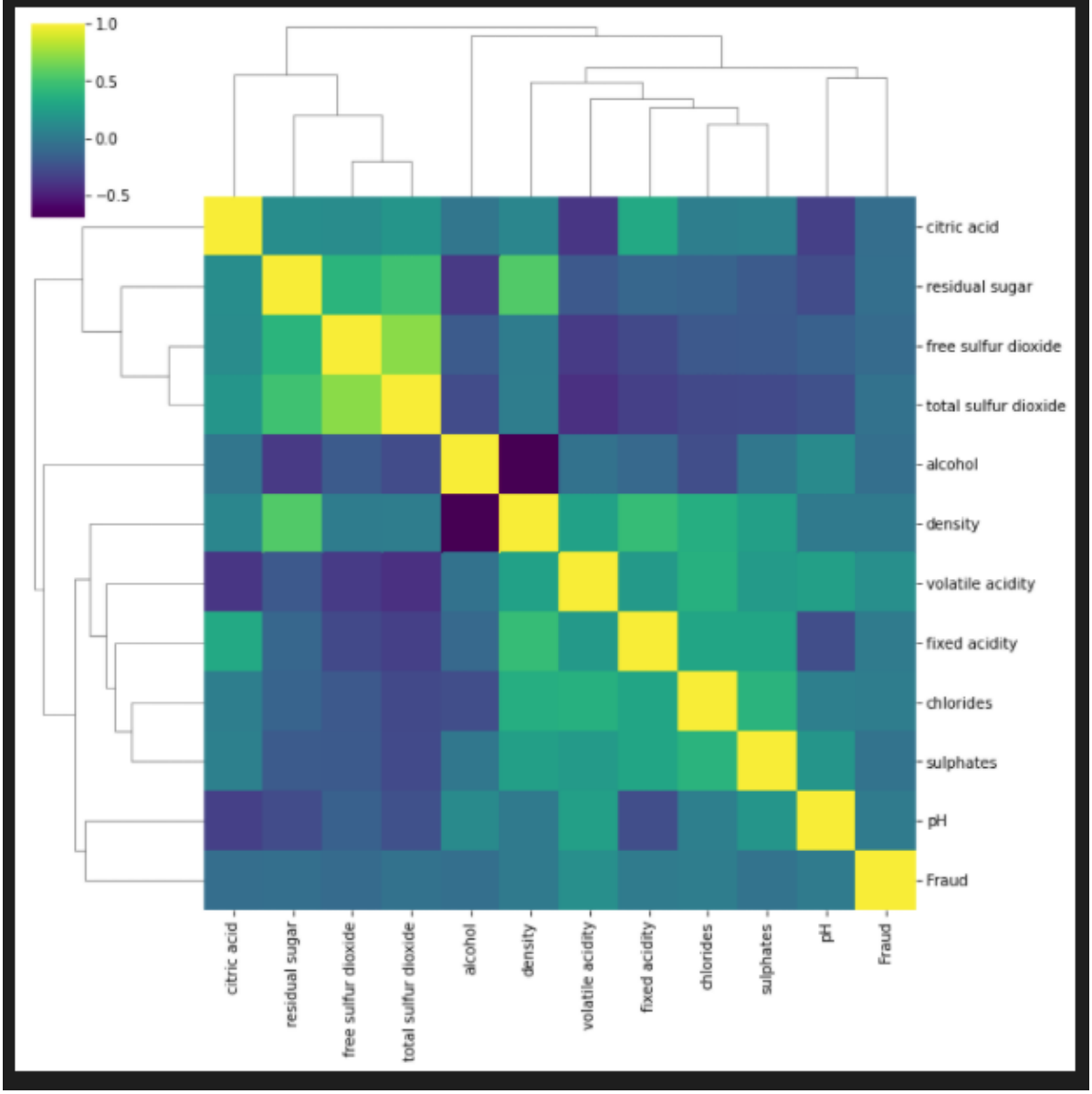
```
<AxesSubplot:>
```



```python
sns.clustermap(df.corr(),cmap="viridis")
# Cluster map birleştirmeli heat maptır.
# Eksen değil df.corr() değeri girilmelidir.
```

✓ 0.6s Python

```
<seaborn.matrix.ClusterGrid at 0x23b26762ac0>
```



ML Model

```python
df["type"] = pd.get_dummies(df["type"],drop_first=True)
# Type verisi string ve red veya white olduğu için nümerik bir
# .... hale getirmek gerekli. bu sebeple dummy variable atandı.
```

✓ 0.4s Python

```python
df = df.drop('Fraud',axis=1)
```

✓ 0.1s Python

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

✓ 0.4s Python

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

✓ 1.2s 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

```
4]
1 from sklearn.preprocessing import StandardScaler
✓ 0.4s

5]
1 scaler = StandardScaler()
✓ 0.3s

6]
1 scaled_X_train = scaler.fit_transform(X_train)
2 scaled_X_test = scaler.transform(X_test)
✓ 0.5s

7]
1 from sklearn.svm import SVC
✓ 0.3s

1]
1 svc = SVC(class_weight="balanced")
2 # class_weight="balanced" : dictionary or balanced
3 # ... fraudlar az olduğu için onlara biraz daha ağırlık
4 # ... verilsin diye böyle bir işlem yapıldı
✓ 0.1s
```

Grid CV

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

✓ 0.1s

```
1  param_grid = {
2      'C':[0.001,0.01,0.1,0.5,1],
3      'gamma':['scale','auto']
4      }
5
6  grid = GridSearchCV(svc,param_grid)
```

✓ 0.8s

```
1  grid.fit(scaled_X_train,y_train)
```

✓ 1m 36.9s

```
GridSearchCV(estimator=SVC(class_weight='balanced'),
             param_grid={'C': [0.001, 0.01, 0.1, 0.5, 1],
                         'gamma': ['scale', 'auto']})
```

```
1  grid.best_params_
```

✓ 0.7s

```
{'C': 1, 'gamma': 'auto'}
```

Evaluation and Results

```
1  sklearn.metrics import confusion_matrix,classification_report
```

✓ 0.1s Pyth

```
1  grid_pred = grid.predict(scaled_X_test)
```

✓ 0.3s Pyth

```
1  confusion_matrix(y_test, grid_pred)
```

✓ 0.1s Pyth

```
array([[ 17,  10],
       [ 92, 531]], dtype=int64)
```

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

✓ 0.4s Pyth

```
              precision    recall  f1-score   support

       Fraud       0.16      0.63      0.25        27
       Legit       0.98      0.85      0.91       623

    accuracy                           0.84       650
   macro avg       0.57      0.74      0.58       650
weighted avg       0.95      0.84      0.88       650
```