


```
from google.colab import files
uploaded = files.upload()
```




Choose files

 WineQT.csv

- **WineQT.csv**(text/csv) - 78057 bytes, last modified: 03/07/2025 - 100% done

Saving WineQT.csv to WineQT.csv

```
from google.colab import files
uploaded = files.upload()
```



Choose files


 WineQT.csv

- **WineQT.csv**(text/csv) - 78057 bytes, last modified: 03/07/2025 - 100% done

Saving WineQT.csv to WineQT (1).csv

```
import pandas as pd

# Use the correct file name as uploaded
df = pd.read_csv('WineQT.csv')
df.head()
```



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulph
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	

Next steps:

Generate code with df

 View recommended plots

New interactive sheet

```
print("Shape:", df.shape)
print("Missing Values:\n", df.isnull().sum())
print("Data Types:\n", df.dtypes)

# Check unique values in target column
print("Quality Counts:\n", df['quality'].value_counts())
```



```

➡ Shape: (1143, 13)
Missing Values:
  fixed acidity          0
  volatile acidity      0
  citric acid           0
  residual sugar        0
  chlorides             0
  free sulfur dioxide   0
  total sulfur dioxide  0
  density               0
  pH                   0
  sulphates            0
  alcohol              0
  quality              0
  Id                   0
dtype: int64
Data Types:
  fixed acidity          float64
  volatile acidity      float64
  citric acid           float64
  residual sugar        float64
  chlorides             float64
  free sulfur dioxide   float64
  total sulfur dioxide  float64
  density               float64
  pH                   float64
  sulphates            float64
  alcohol              float64
  quality              int64
  Id                   int64
dtype: object
Quality Counts:
  quality
5    483
6    462
7    143
4     33
8     16
3      6
Name: count, dtype: int64

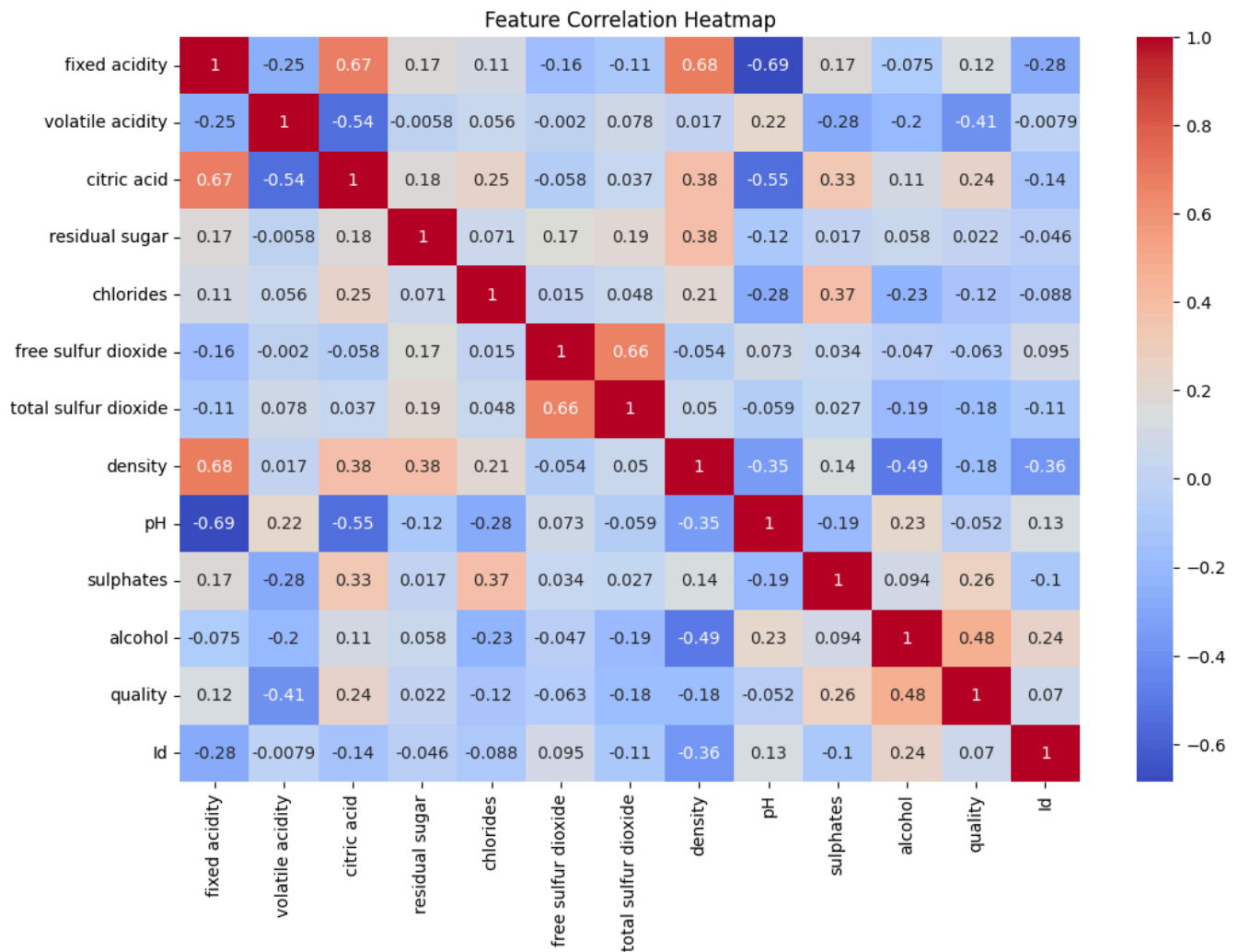
```

```

plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Feature Correlation Heatmap")
plt.show()

```





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

```
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
)
```

```
#random forest classifier
```

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

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
rf = RandomForestClassifier(random_state=42)
```

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

```
y_pred_rf = rf.predict(X_test)
```

```
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Classification Report:\n", classification_report(y_test, y_pred_rf))
```



Random Forest Accuracy: 0.6812227074235808

Classification Report:

	precision	recall	f1-score	support
4	0.00	0.00	0.00	6
5	0.73	0.75	0.74	96
6	0.63	0.70	0.66	99
7	0.75	0.58	0.65	26
8	0.00	0.00	0.00	2
accuracy			0.68	229
macro avg	0.42	0.40	0.41	229
weighted avg	0.66	0.68	0.67	229

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedWarning:
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedWarning:
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedWarning:
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
#support vector classifier
from sklearn.svm import SVC
```

```
svc = SVC()
svc.fit(X_train, y_train)
y_pred_svc = svc.predict(X_test)
```

```
print("SVC Accuracy:", accuracy_score(y_test, y_pred_svc))
```



SVC Accuracy: 0.49344978165938863

```
#SGD Classifier
from sklearn.linear_model import SGDClassifier
```

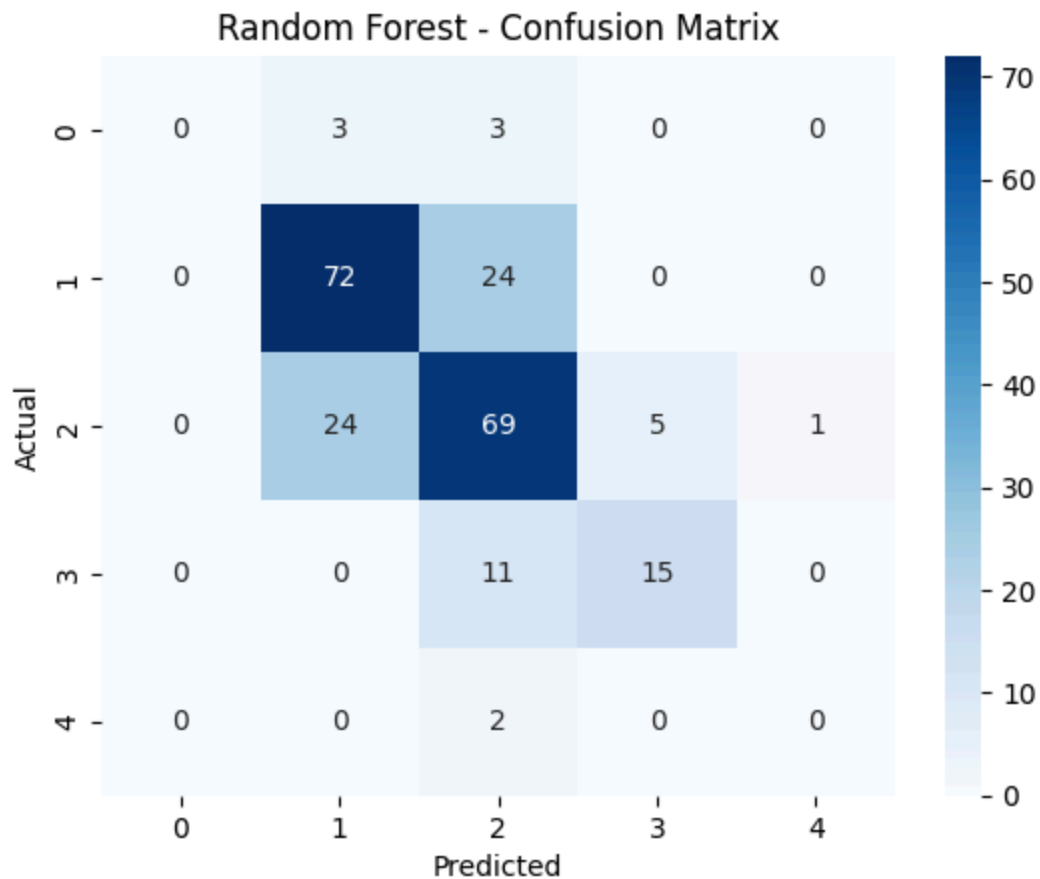
```
sgd = SGDClassifier(loss='hinge', max_iter=1000, random_state=42)
sgd.fit(X_train, y_train)
y_pred_sgd = sgd.predict(X_test)
```

```
print("SGD Accuracy:", accuracy_score(y_test, y_pred_sgd))
```



SGD Accuracy: 0.5327510917030568

```
#confusion matrix
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d', cmap='Blues')
plt.title("Random Forest - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



Start coding or [generate](#) with AI.