**Wine quality classification using svm and Random forest algorithm**

# Step 0: Import libraries

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, f1\_score, recall\_score, precision\_score

import plotly.graph\_objects as go

# Step 1: Load the saved CSV and fix the formatting

df = pd.read\_csv("winequality-red.csv")

# If only one column exists, it's a malformed CSV – fix it

if df.shape[1] == 1:

df = df.iloc[:, 0].str.split(",", expand=True)

# Assign proper column names manually

df.columns = [

'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides',

'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates',

'alcohol', 'quality'

]

# Convert all values to numeric

df = df.apply(pd.to\_numeric)

# Step 2: Visualize correlation heatmap

plt.figure(figsize=(10, 6))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.title("📊 Correlation Heatmap - Wine Quality Features")

plt.show()

# Step 3: Basic statistics

stats = df.describe().T[['mean', 'std']]

stats['median'] = df.median()

stats['variance'] = df.var()

print("\n📈 Preprocessed Statistics (Top 5 features):")

print(stats[['mean', 'median', 'variance', 'std']].head())

# Step 4: Prepare for classification

df['quality\_label'] = df['quality'].apply(lambda x: 1 if x >= 7 else 0)

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

print("\n📋 Preprocessed Data Table (first 10 rows):")

print(df.head(10))

import plotly.express as px

# Step 4.1: Innovative input data visualization - Parallel Coordinates Plot

fig = px.parallel\_coordinates(

df,

color="quality\_label",

dimensions=[

'fixed acidity', 'volatile acidity', 'citric acid',

'residual sugar', 'chlorides', 'free sulfur dioxide',

'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol'

],

color\_continuous\_scale=px.colors.diverging.Tealrose,

labels={"quality\_label": "Quality Label"},

title="🍷 Parallel Coordinates Plot - Input Feature Distributions by Wine Quality"

)

fig.show()

X = df.drop('quality\_label', axis=1)

y = df['quality\_label']

# Scale features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# Step 5: Train models

svm = SVC()

svm.fit(X\_train, y\_train)

svm\_preds = svm.predict(X\_test)

rf = RandomForestClassifier()

rf.fit(X\_train, y\_train)

rf\_preds = rf.predict(X\_test)

# Step 6: Evaluation function

def evaluate(y\_true, y\_pred):

return {

'Accuracy': accuracy\_score(y\_true, y\_pred),

'F1 Score': f1\_score(y\_true, y\_pred),

'Recall': recall\_score(y\_true, y\_pred),

'Precision': precision\_score(y\_true, y\_pred)

}

svm\_scores = evaluate(y\_test, svm\_preds)

rf\_scores = evaluate(y\_test, rf\_preds)

print("\n🔍 SVM Performance:")

for metric, val in svm\_scores.items():

print(f"{metric}: {val:.2f}")

print("\n🔍 Random Forest Performance:")

for metric, val in rf\_scores.items():

print(f"{metric}: {val:.2f}")

# Step 7: Radial Performance Graph

metrics = list(svm\_scores.keys())

svm\_values = list(svm\_scores.values())

rf\_values = list(rf\_scores.values())

fig = go.Figure()

fig.add\_trace(go.Barpolar(

r=svm\_values,

theta=metrics,

name='SVM',

marker\_color='navy',

opacity=0.7

))

fig.add\_trace(go.Barpolar(

r=rf\_values,

theta=metrics,

name='Random Forest',

marker\_color='darkred',

opacity=0.7

))

fig.update\_layout(

title="🍷 Radial Comparison - SVM vs Random Forest on Wine Quality",

polar=dict(radialaxis=dict(visible=True, range=[0, 1])),

showlegend=True

)

fig.show()







