SENTIMENT ANALYSIS OF TWEETS

A Project Report on

SENTIMENT ANALYSIS OF TWEETS

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science Engineering

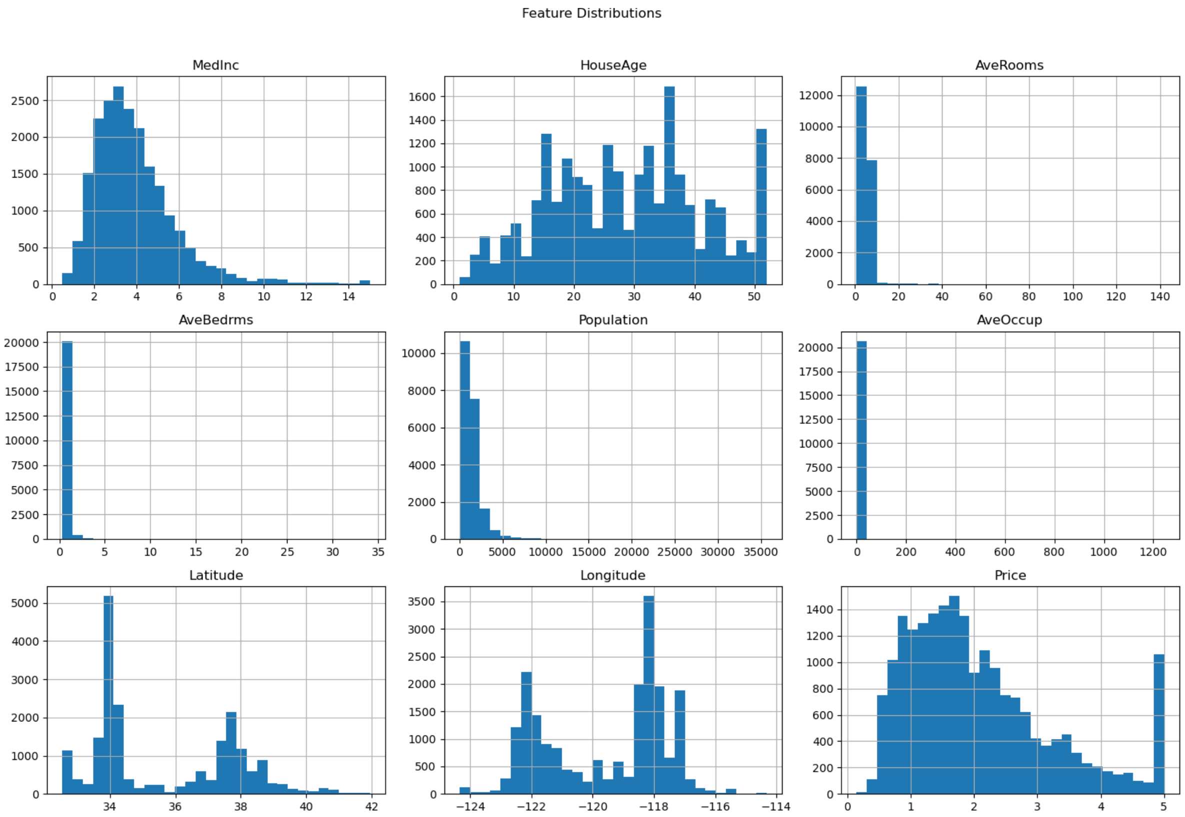
BY

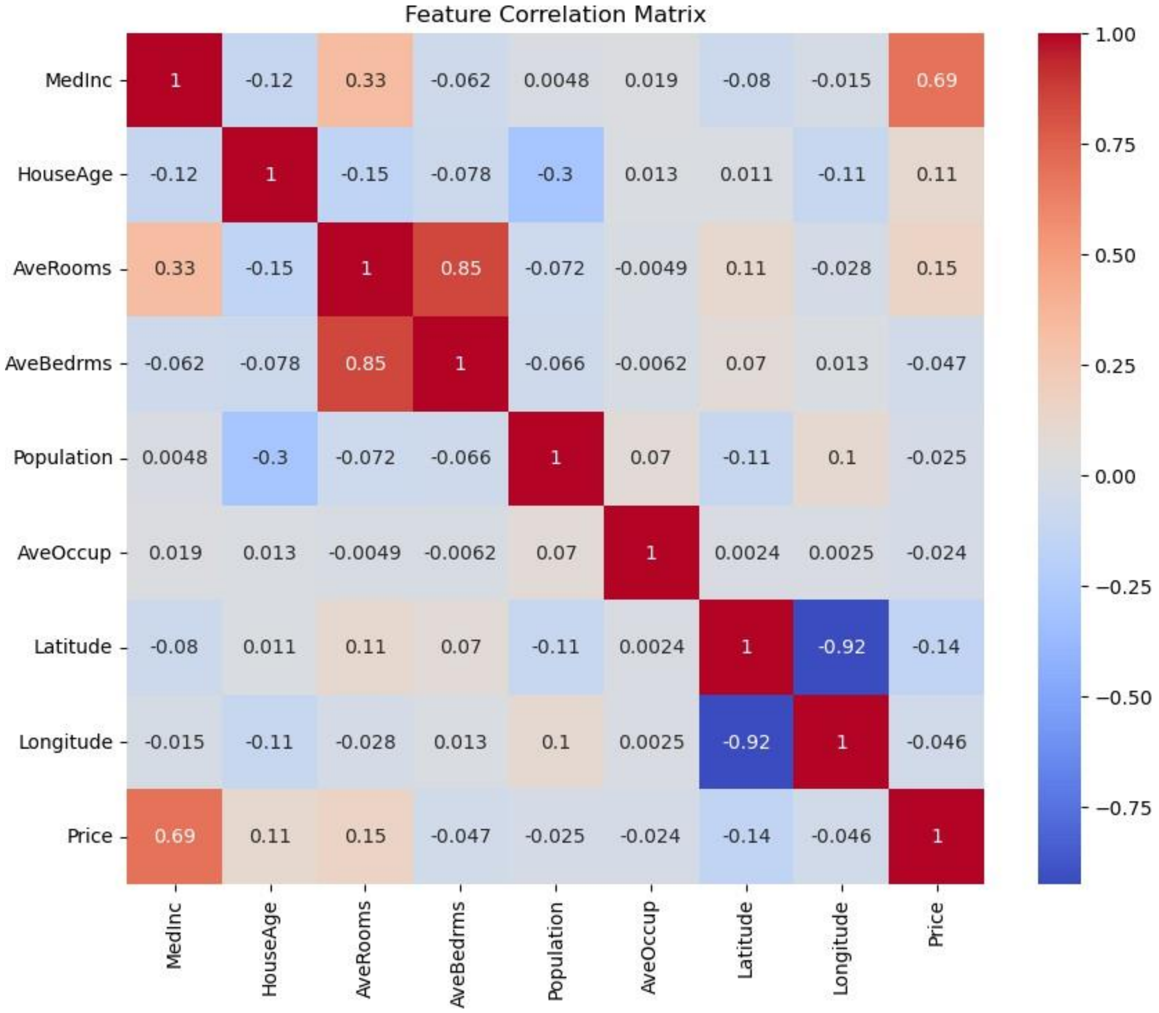
KONDAL SRINIDHI SAGAR

```
# House Price Prediction (Supervised Learning – Regression)
# Predict home prices using ML models on California Housing Dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
import pickle

# ---------------------------
# 1. Load and Explore the Data
# ---------------------------
print("Loading California housing data...")
housing = fetch_california_housing()
df = pd.DataFrame(housing.data, columns=housing.feature_names)
df['Price'] = housing.target

print("\nFirst few rows of data:")
print(df.head())

print(f"\nDataset shape: {df.shape}")
print(f"Price range: ${df['Price'].min()*100000:,.2f} - ${df['Price'].max()*1000

# ---------------------------
# 2. Visualize the Data
# ---------------------------
print("\nVisualizing data...")
plt.figure(figsize=(12, 8))
df.hist(bins=30, figsize=(15, 10))
plt.suptitle("Feature Distributions", y=1.02)
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Feature Correlation Matrix")
plt.show()

# ---------------------------
# 3. Feature Selection & Preprocessing
# ---------------------------
# Drop Latitude and Longitude for simplicity
X = df.drop(['Price', 'Latitude', 'Longitude'], axis=1)
y = df['Price']

# ---------------------------
# 4. Train/Test Split
# ---------------------------
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```python
# ---------------------------
# 5. Train Models
# ---------------------------
models = {
    "Linear Regression": LinearRegression(),
    "Random Forest": RandomForestRegressor(random_state=42),
    "XGBoost": XGBRegressor(random_state=42)
}

results = []

for name, model in models.items():
    print(f"\nTraining {name}...")
    model.fit(X_train, y_train)
    preds = model.predict(X_test)

    r2 = r2_score(y_test, preds)
    mae = mean_absolute_error(y_test, preds)
    rmse = np.sqrt(mean_squared_error(y_test, preds))

    results.append({
        "Model": name,
        "R² Score": r2,
        "MAE ($)": mae * 100000,
        "RMSE ($)": rmse * 100000
    })

    print(f"R²: {r2:.3f}, MAE: ${mae*100000:,.2f}, RMSE: ${rmse*100000:,.2f}")

# ---------------------------
# 6. Model Evaluation
# ---------------------------
print("\nModel Evaluation Summary:")
results_df = pd.DataFrame(results)
print(results_df.sort_values("R² Score", ascending=False))

# ---------------------------
# 7. Save Best Model
# ---------------------------
best_model = XGBRegressor(random_state=42)
best_model.fit(X_train, y_train)

with open("california_model.pkl", "wb") as f:
    pickle.dump(best_model, f)
print("\nBest model saved as 'california_model.pkl'")

# ---------------------------
# 8. Streamlit Web App (Optional)
# ---------------------------
def run_web_app():
    import streamlit as st

    st.title("California House Price Predictor")
    st.write("Estimate house price based on selected features.")

    st.sidebar.header("Input Features")
    MedInc = st.sidebar.slider("Median Income (in $10,000)", 0.0, 15.0, 3.0)
    HouseAge = st.sidebar.slider("House Age", 0, 100, 30)
    AveRooms = st.sidebar.slider("Average Rooms", 1.0, 15.0, 5.0)
    AveBedrms = st.sidebar.slider("Average Bedrooms", 0.5, 5.0, 1.0)
    Population = st.sidebar.slider("Block Population", 0, 10000, 1500)
    AveOccup = st.sidebar.slider("Average Occupancy", 1.0, 10.0, 3.0)

    if st.sidebar.button("Predict Price"):
        input_features = [[MedInc, HouseAge, AveRooms, AveBedrms, Population, Av
        with open("california_model.pkl", "rb") as f:
            model = pickle.load(f)
        prediction = model.predict(input_features)[0]
        st.success(f"Estimated House Price: ${prediction*100000:,.2f}")

# To run the web app:
# 1. Save this script as app.py
# 2. Run: streamlit run app.py
# run_web_app()  # Uncomment to launch

# ---------------------------
# End
# ---------------------------
print("\nYou can now:")
print("- Use the saved model for predictions")
print("- Run run_web_app() to launch a Streamlit app")
```

```
Loading California housing data...

First few rows of data:
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88   
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86   
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85   
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85   
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85   

   Longitude  Price  
0    -122.23  4.526  
1    -122.22  3.585  
2    -122.24  3.521  
3    -122.25  3.413  
4    -122.25  3.422  

Dataset shape: (20640, 9)
Price range: $14,999.00 - $500,001.00

Visualizing data...
<Figure size 1200x800 with 0 Axes>
```

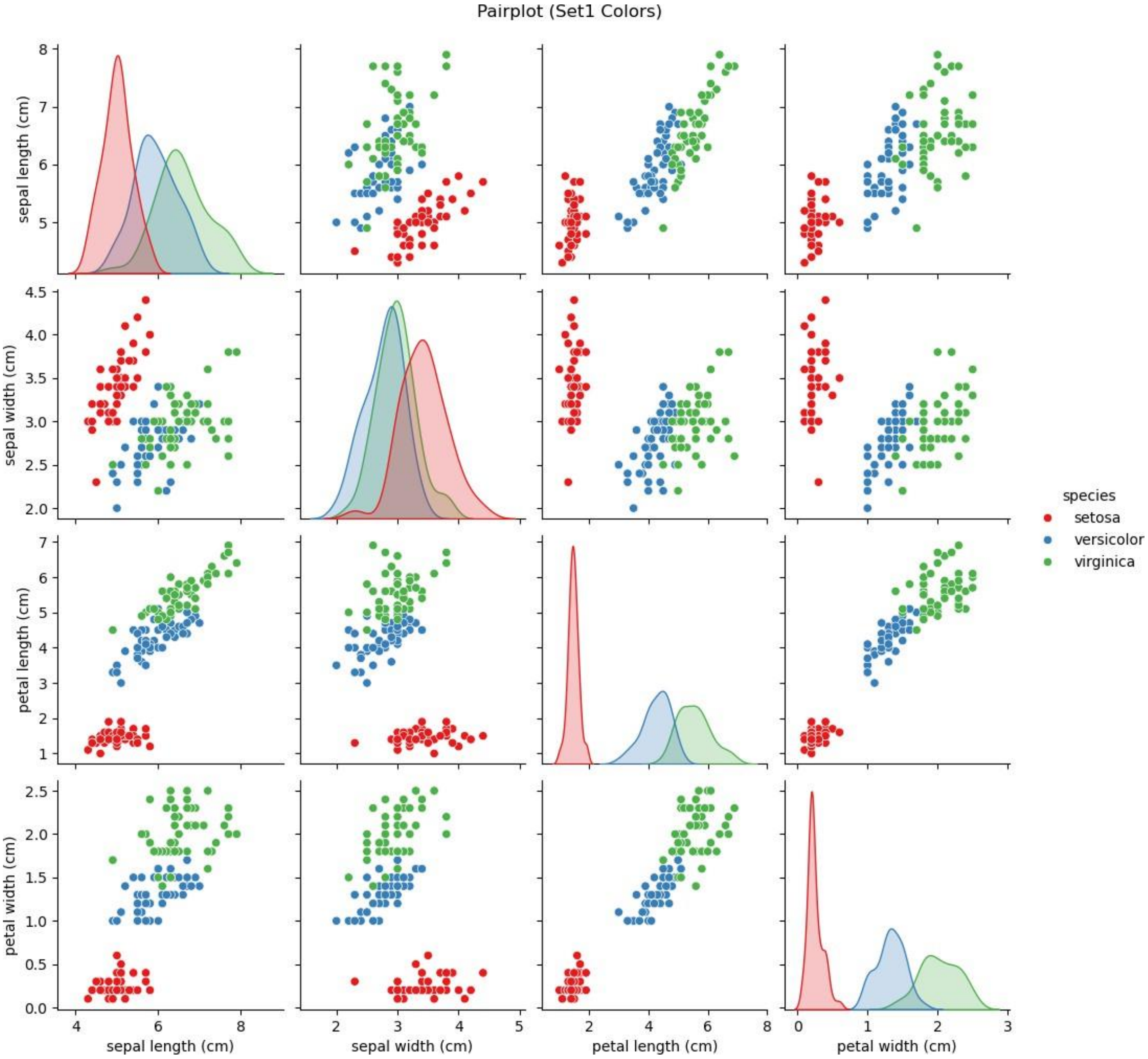

Feature Distributions
MedInc
0 2 4 6 8 10 12 14
0 500 1000 1500 2000 2500
HouseAge
0 10 20 30 40 50
0 200 400 600 800 1000 1200 1400 1600
AveRooms
0 20 40 60 80 100 120 140
0 2000 4000 6000 8000 10000 12000
AveBedrms
0 5 10 15 20 25 30 35
0 2500 5000 7500 10000 12500 15000 17500 20000
Population
0 5000 10000 15000 20000 25000 30000 35000
0 2000 4000 6000 8000 10000
AveOccup
0 200 400 600 800 1000 1200
0 2500 5000 7500 10000 12500 15000 17500 20000
Latitude
34 36 38 40 42
0 1000 2000 3000 4000 5000
Longitude
−124 −122 −120 −118 −116 −114
0 500 1000 1500 2000 2500 3000 3500
Price
0 1 2 3 4 5
0 200 400 600 800 1000 1200 1400


Feature Correlation Matrix
MedInc 1 -0.12 0.33 -0.062 0.0048 0.019 -0.08 -0.015 0.69
HouseAge -0.12 1 -0.15 -0.078 -0.3 0.013 0.011 -0.11 0.11
AveRooms 0.33 -0.15 1 0.85 -0.072 -0.0049 0.11 -0.028 0.15
AveBedrms -0.062 -0.078 0.85 1 -0.066 -0.0062 0.07 0.013 -0.047
Population 0.0048 -0.3 -0.072 -0.066 1 0.07 -0.11 0.1 -0.025
AveOccup 0.019 0.013 -0.0049 -0.0062 0.07 1 0.0024 0.0025 -0.024
Latitude -0.08 0.011 0.11 0.07 -0.11 0.0024 1 -0.92 -0.14
Longitude -0.015 -0.11 -0.028 0.013 0.1 0.0025 -0.92 1 -0.046
Price 0.69 0.11 0.15 -0.047 -0.025 -0.024 -0.14 -0.046 1
MedInc HouseAge AveRooms AveBedrms Population AveOccup Latitude Longitude Price
1.00 0.75 0.50 0.25 0.00 −0.25 −0.50 −0.75

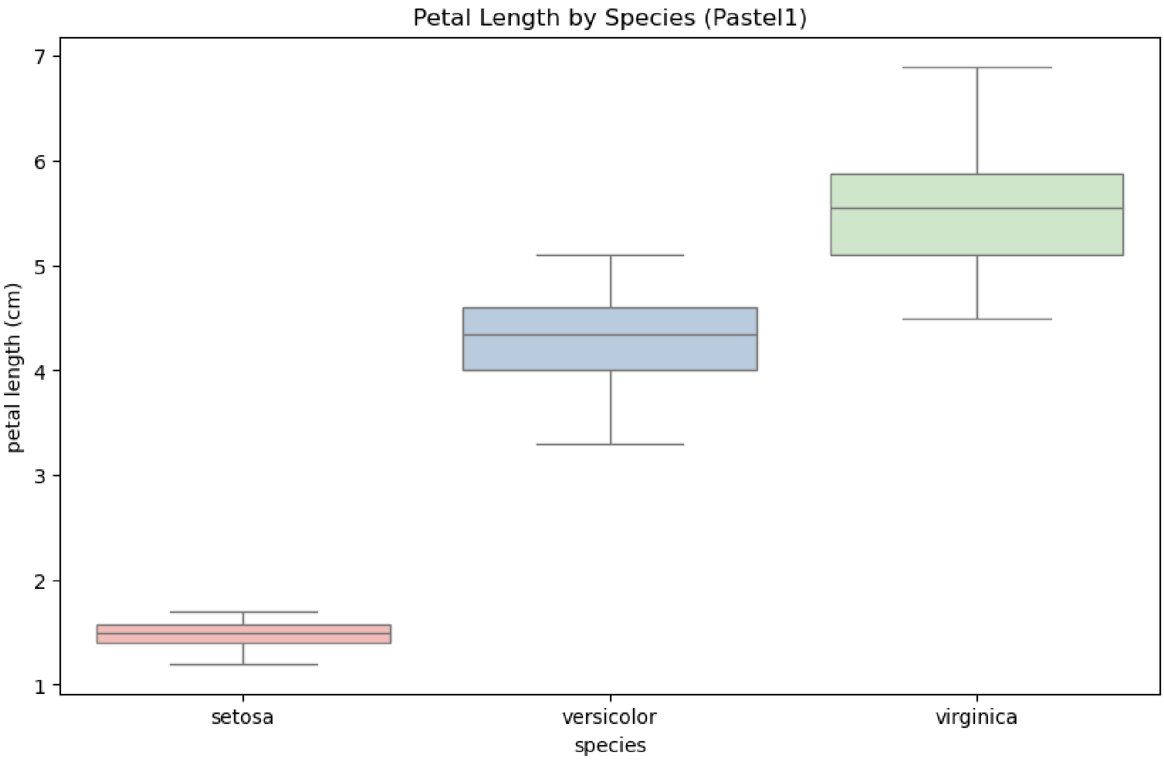
```
Training Linear Regression...
R²: 0.510, MAE: $57,921.41, RMSE: $80,136.59

Training Random Forest...
R²: 0.678, MAE: $46,191.21, RMSE: $65,005.54

Training XGBoost...
R²: 0.671, MAE: $46,800.32, RMSE: $65,639.16

Model Evaluation Summary:
               Model  R² Score       MAE ($)      RMSE ($)
1      Random Forest  0.677527  46191.207871  65005.544472
2            XGBoost  0.671210  46800.316097  65639.155268
0  Linear Regression  0.509934  57921.406655  80136.585369

Best model saved as 'california_model.pkl'

You can now:
- Use the saved model for predictions
- Run run_web_app() to launch a Streamlit app
```

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep

# 1. Load the dataset
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)

# 2. Visualize - Pairplot with Set1 palette
print("📊 Creating pairplot with Set1
sns.pairplot(df, hue="species", palette="Set1")
plt.suptitle("Pairplot (Set1 Colors)", y=1.02)
plt.show()

# 3. Visualize - Boxplot with Pastel1 palette (with hue to prevent deprecation w
print("📊 Creating boxplot with Pastel1
plt.figure(figsize=(10, 6))
sns.boxplot(x="species", y="petal length (cm)", data=df, hue="species", palette=
plt.title("Petal Length by Species (Pastel1)")
plt.show()

# 4. Visualize - Violin plot with Dark2 palette
print("📊 Creating violin plot with Dark2
plt.figure(figsize=(10, 6))
sns.violinplot(x="species", y="sepal width (cm)", data=df, hue="species", palett
plt.title("Sepal Width by Species (Dark2)")
plt.show()

# 5. Prepare features and labels
X = df.drop("species", axis=1)
y = df["species"]

# 6. Split the data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# 7. Train the model
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)

# 8. Evaluate the model
y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)

print(f"\n✔ Accuracy: {acc:.2%}")
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

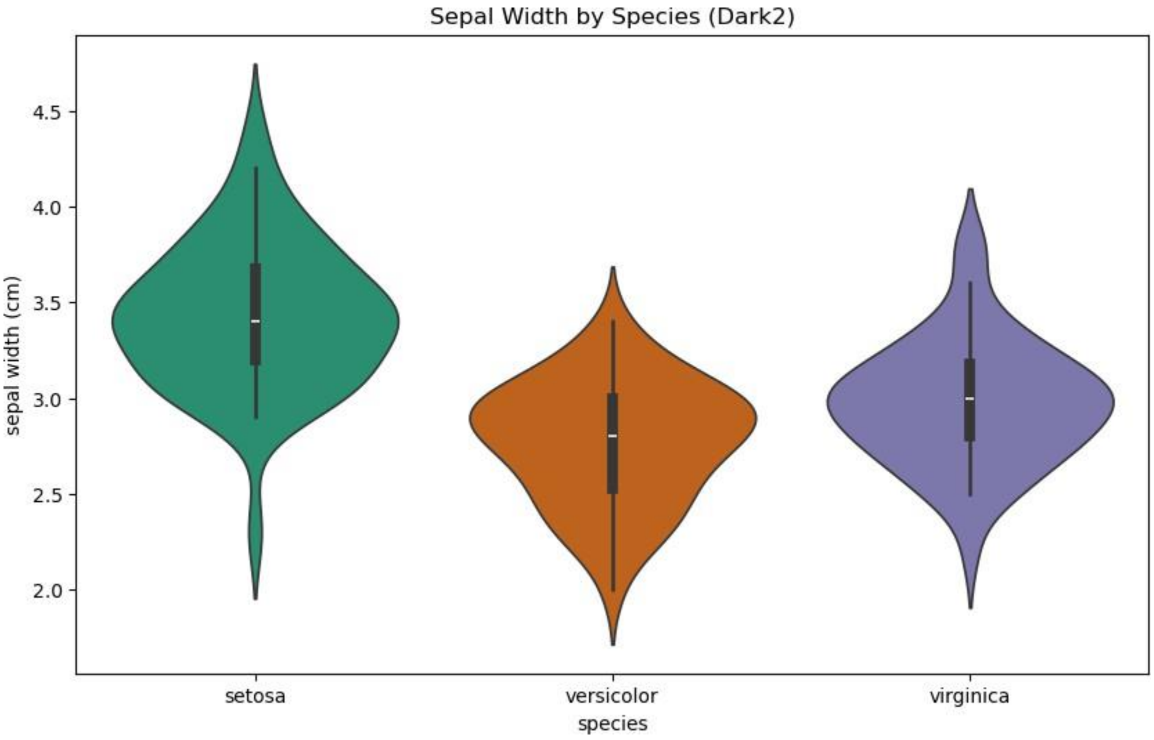
```
Creating pairplot with Set1 palette...
```



```
Creating boxplot with Pastel1 palette...
```



```
Creating violin plot with Dark2 palette...
```



```
✔ Accuracy: 100.00%

Confusion Matrix:
[[10  0  0]
 [ 0 10  0]
 [ 0  0 10]]

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00        10
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```