Data and Artificial Intelligence
Cyber Shujaa Program

Week 8 Assignment
Classification Models

Student Name: Shirleen Nanetia Simon

Student ID: CS-DA01-25077

Introduction

The purpose of this assignment was to apply supervised machine learning classification techniques to the Wine dataset from scikit-learn. The goal was to build and evaluate several models, analyze their performance using standard metrics, and determine which model performs best under similar conditions. This hands-on exercise reinforced practical skills in data preprocessing, exploratory data analysis (EDA), model implementation, and evaluation.

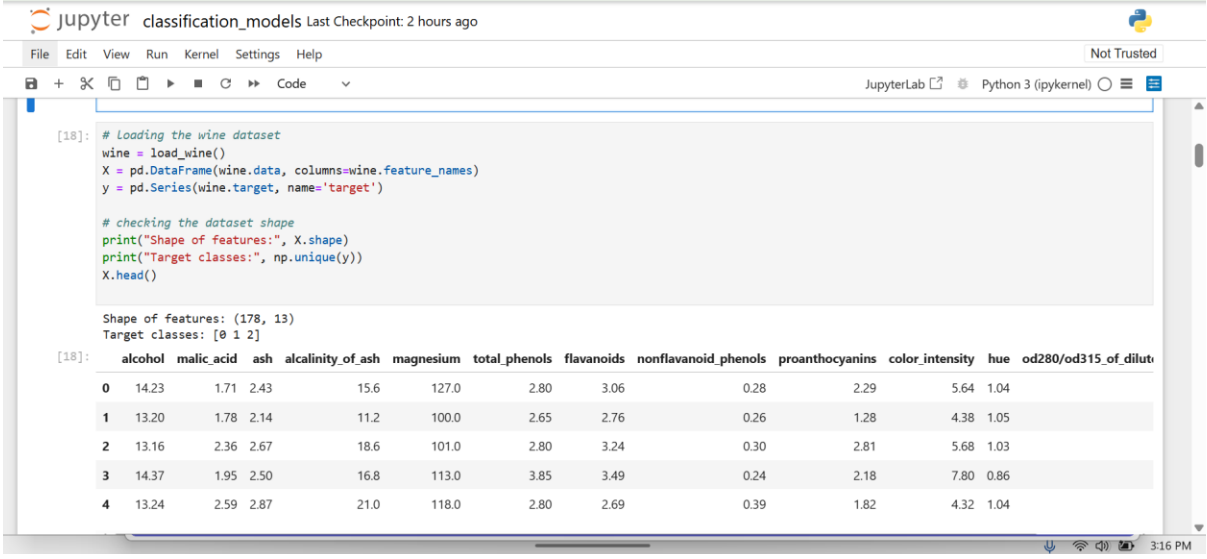
The following classification models were implemented:

- Logistic Regression
- Decision Tree
- Random Forest
- k-Nearest Neighbors (KNN)
- Naive Bayes
- Support Vector Machine (SVM)

Tasks Completed

1. Dataset Loading and Exploration

- The Wine dataset was loaded from scikit-learn.
- The dataset contains **178 samples** with **13 features** and a target variable representing three wine classes.



jupyter classification_models Last Checkpoint: 2 hours ago
File Edit View Run Kernel Settings Help
Not Trusted
Code
JupyterLab
Python 3 (ipykernel)
3:16 PM

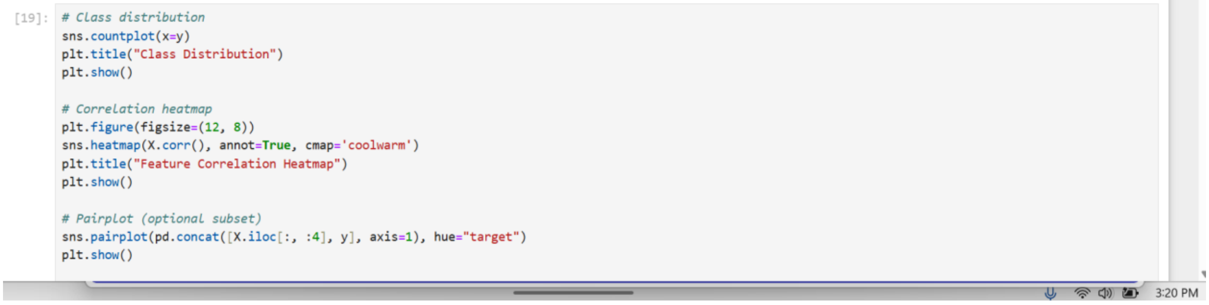
```
[18]: # Loading the wine dataset
wine = load_wine()
X = pd.DataFrame(wine.data, columns=wine.feature_names)
y = pd.Series(wine.target, name='target')

# checking the dataset shape
print("Shape of features:", X.shape)
print("Target classes:", np.unique(y))
X.head()
```

```
Shape of features: (178, 13)
Target classes: [0 1 2]
```

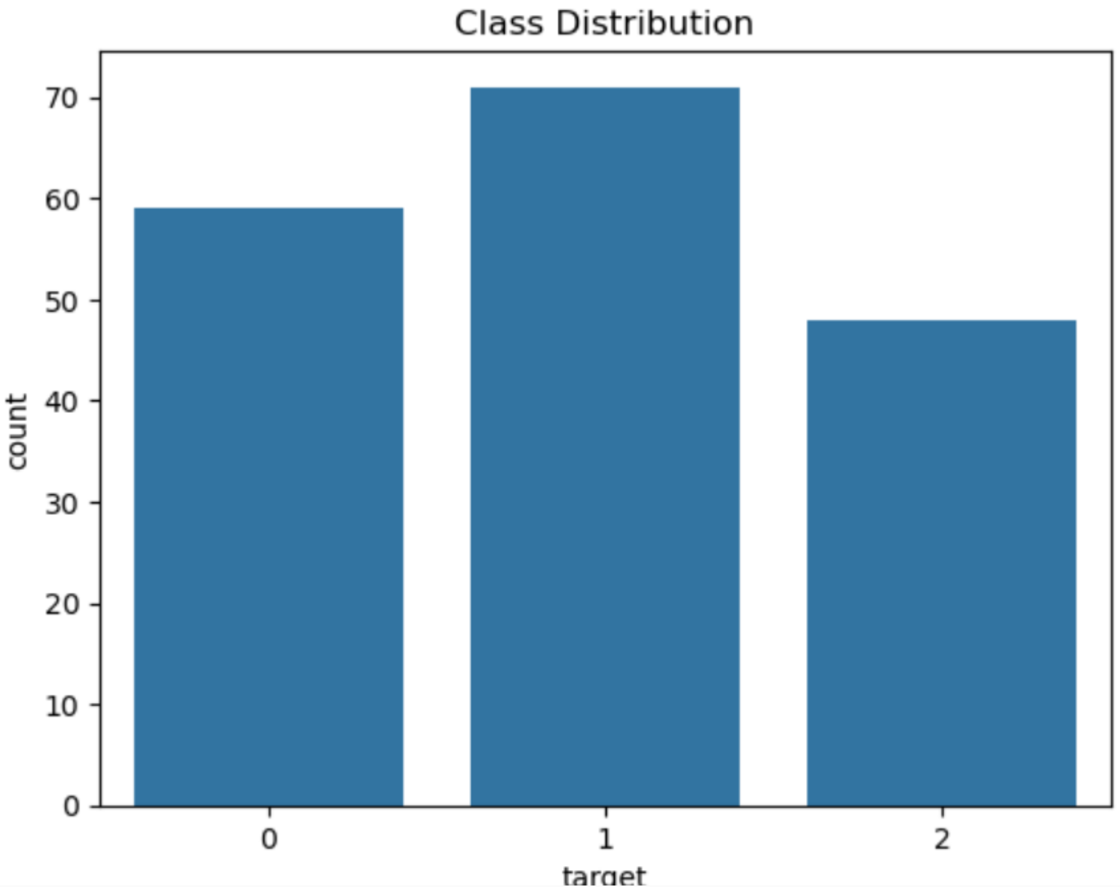
[18]:

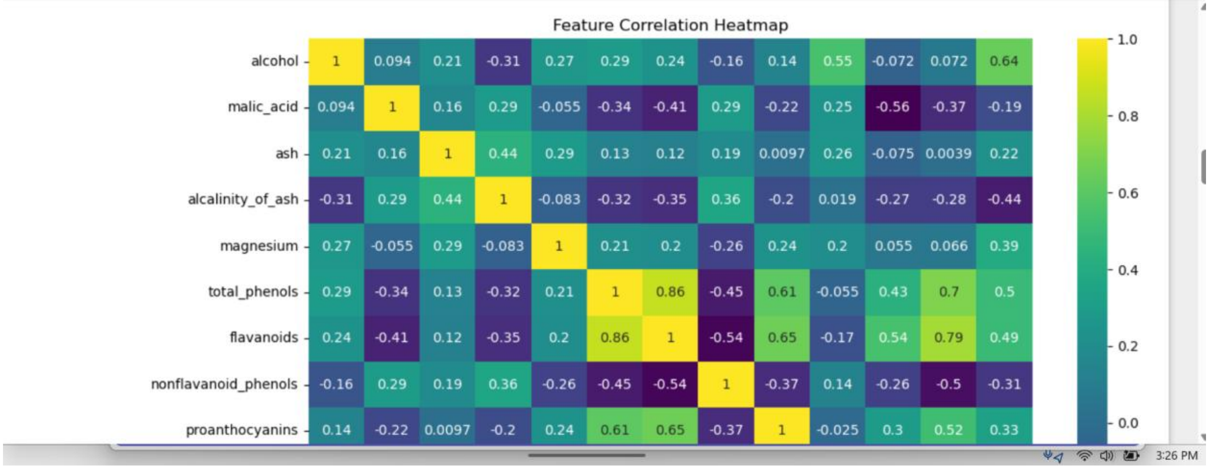
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_dilut
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	

- Exploratory Data Analysis (EDA) was performed:
  - Checked for missing values and found none.
  - Visualized feature distributions using histograms and pair plots.
  - Analyzed the correlation matrix to understand relationships between variables.
  - Observed class distributions to confirm a balanced dataset

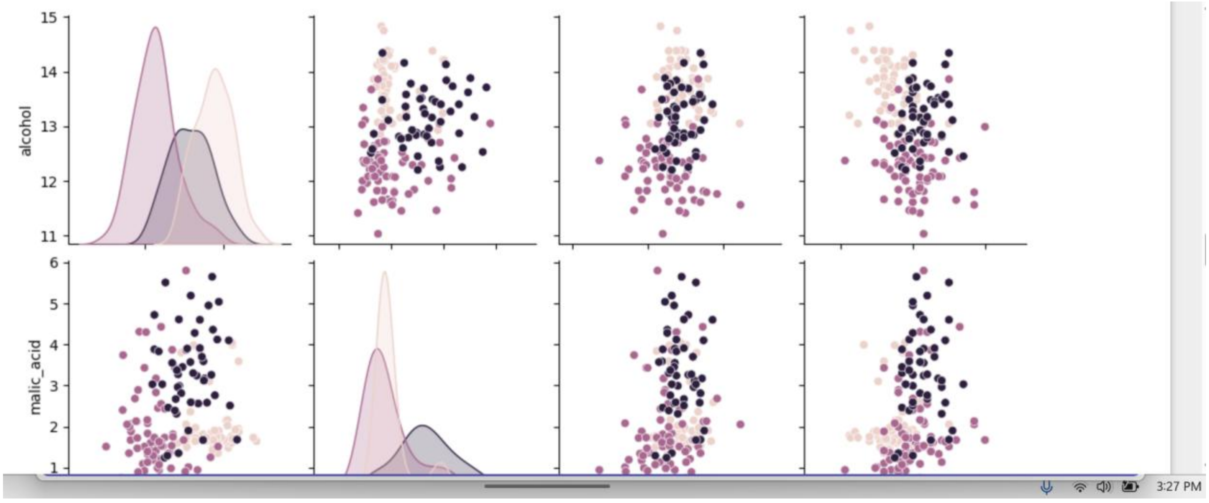


```
[19]: # Class distribution
sns.countplot(x=y)
plt.title("Class Distribution")
plt.show()

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

# Pairplot (optional subset)
sns.pairplot(pd.concat([X.iloc[:, :4], y], axis=1), hue="target")
plt.show()
```



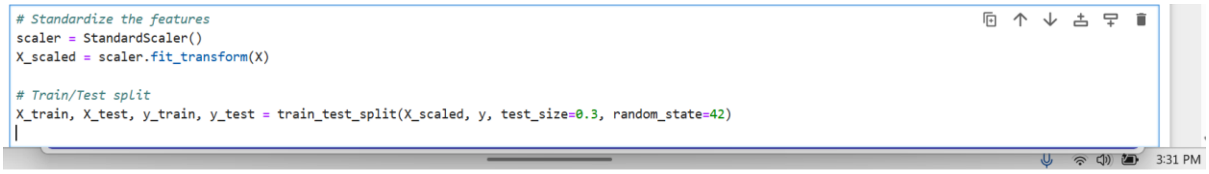

Feature Correlation Heatmap

alcohol	1	0.094	0.21	-0.31	0.27	0.29	0.24	-0.16	0.14	0.55	-0.072	0.072	0.64
malic_acid	0.094	1	0.16	0.29	-0.055	-0.34	-0.41	0.29	-0.22	0.25	-0.56	-0.37	-0.19
ash	0.21	0.16	1	0.44	0.29	0.13	0.12	0.19	0.0097	0.26	-0.075	0.0039	0.22
alcalinity_of_ash	-0.31	0.29	0.44	1	-0.083	-0.32	-0.35	0.36	-0.2	0.019	-0.27	-0.28	-0.44
magnesium	0.27	-0.055	0.29	-0.083	1	0.21	0.2	-0.26	0.24	0.2	0.055	0.066	0.39
total_phenols	0.29	-0.34	0.13	-0.32	0.21	1	0.86	-0.45	0.61	-0.055	0.43	0.7	0.5
flavanoids	0.24	-0.41	0.12	-0.35	0.2	0.86	1	-0.54	0.65	-0.17	0.54	0.79	0.49
nonflavanoid_phenols	-0.16	0.29	0.19	0.36	-0.26	-0.45	-0.54	1	-0.37	0.14	-0.26	-0.5	-0.31
proanthocyanins	0.14	-0.22	0.0097	-0.2	0.24	0.61	0.65	-0.37	1	-0.025	0.3	0.52	0.33



2. Data Preparation

- Features and target variables were separated.
- The data was split into **training and testing sets (70:30 split)** using train_test_split.
- Feature scaling was applied where necessary, especially for models sensitive to feature magnitudes (e.g., Logistic Regression, KNN, and SVM).



```
# Standardize the 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=0.3, random_state=42)
```

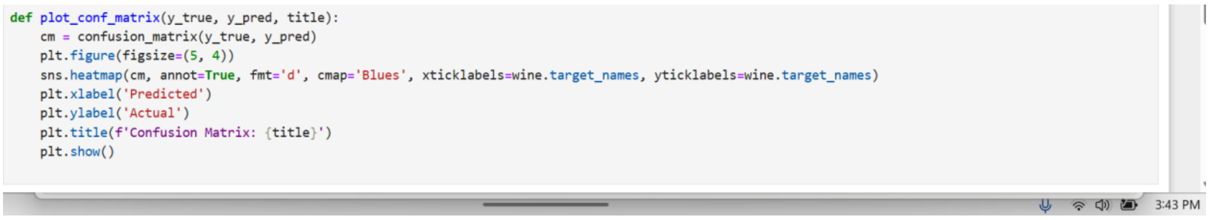
3. Model Building , Training and Evaluation

Six classification algorithms were trained on the Wine dataset:

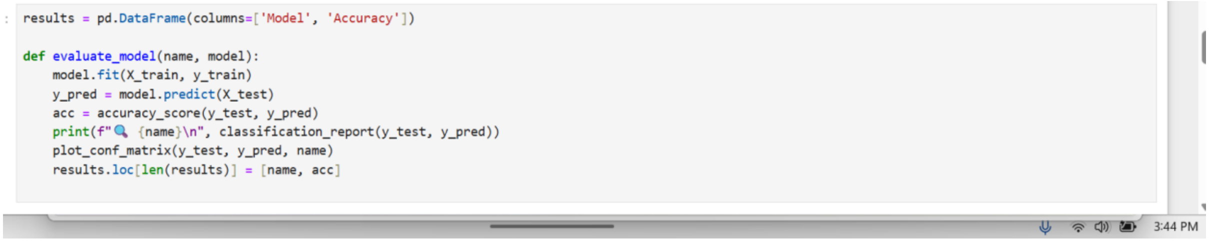
- **Logistic Regression:** A linear model suitable for multiclass problems.
- **Decision Tree Classifier:** A non-linear model capable of capturing complex patterns.
- **Random Forest Classifier:** An ensemble of decision trees to improve accuracy and reduce overfitting.
- **K-Nearest Neighbors (KNN):** A distance-based classifier relying on neighboring data points.
- **Naive Bayes Classifier:** A probabilistic model assuming feature independence.
- **Support Vector Machine (SVM):** A classifier maximizing margins between classes.

For each model, the following metrics were computed:

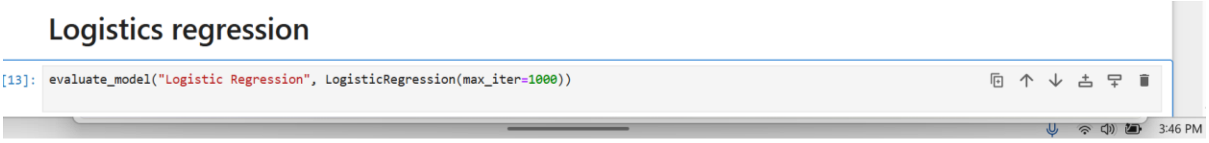
- **Accuracy Score:** Overall correctness of predictions.
- **Classification Report:** Precision, recall, and F1-score per class.
- **Confusion Matrix:** Detailed comparison of true vs. predicted values, visualized using heatmaps.



```
def plot_conf_matrix(y_true, y_pred, title):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(5, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=wine.target_names, yticklabels=wine.target_names)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f'Confusion Matrix: {title}')
    plt.show()
```

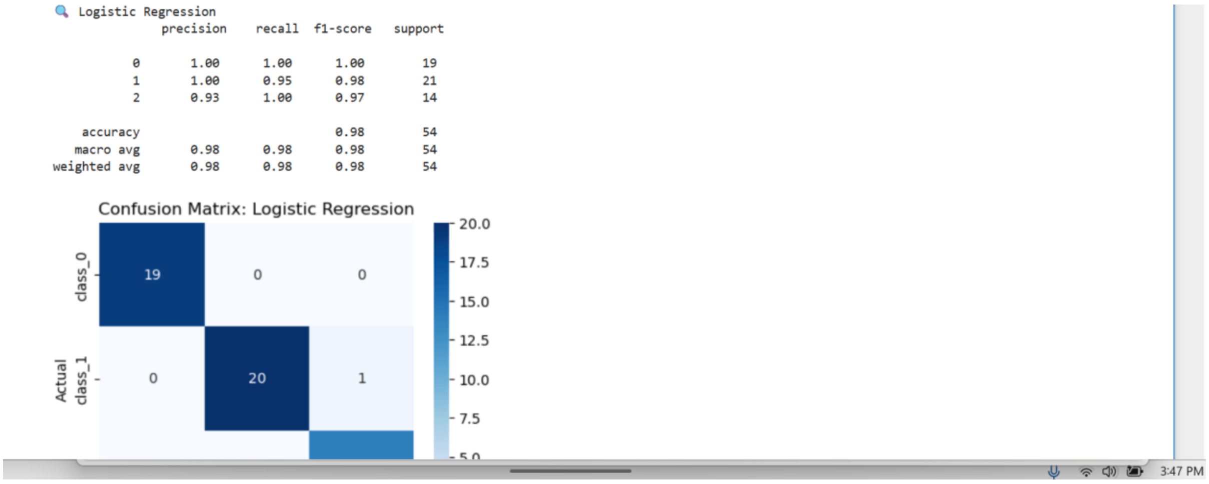


```
results = pd.DataFrame(columns=['Model', 'Accuracy'])

def evaluate_model(name, model):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"🔍 {name}\n", classification_report(y_test, y_pred))
    plot_conf_matrix(y_test, y_pred, name)
    results.loc[len(results)] = [name, acc]
```



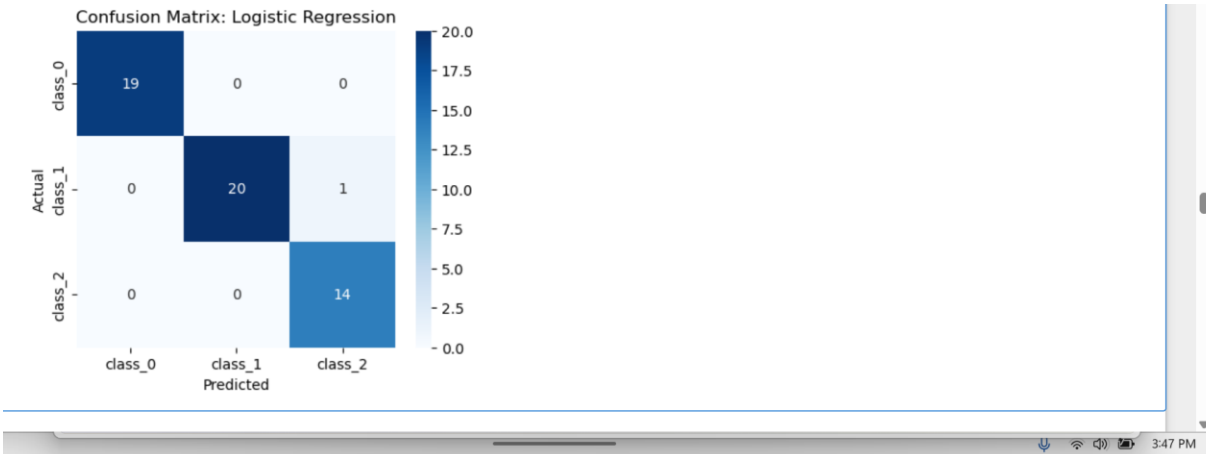
Logistics regression

```
[13]: evaluate_model("Logistic Regression", LogisticRegression(max_iter=1000))
```

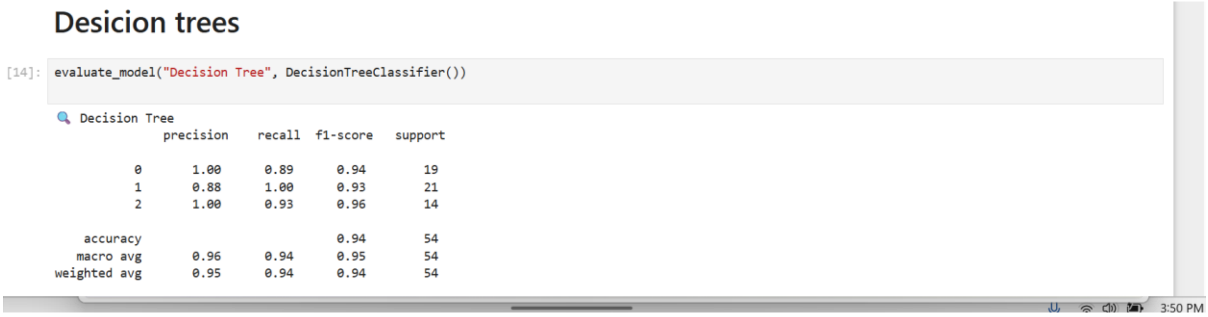


```
🔍 Logistic Regression
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.95      0.98        21
           2       0.93      1.00      0.97        14

    accuracy                           0.98        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54
```

Confusion Matrix: Logistic Regression



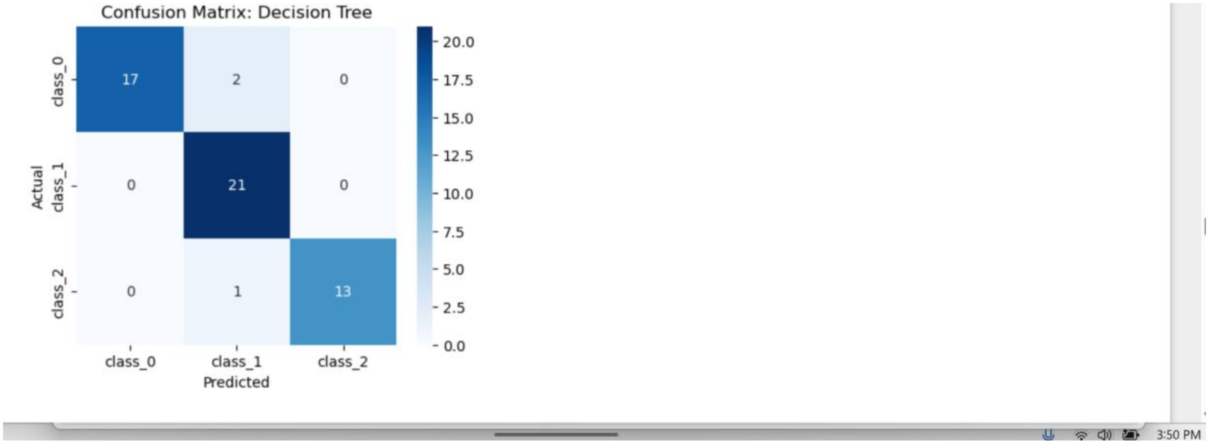
Confusion Matrix: Logistic Regression



Desicion trees

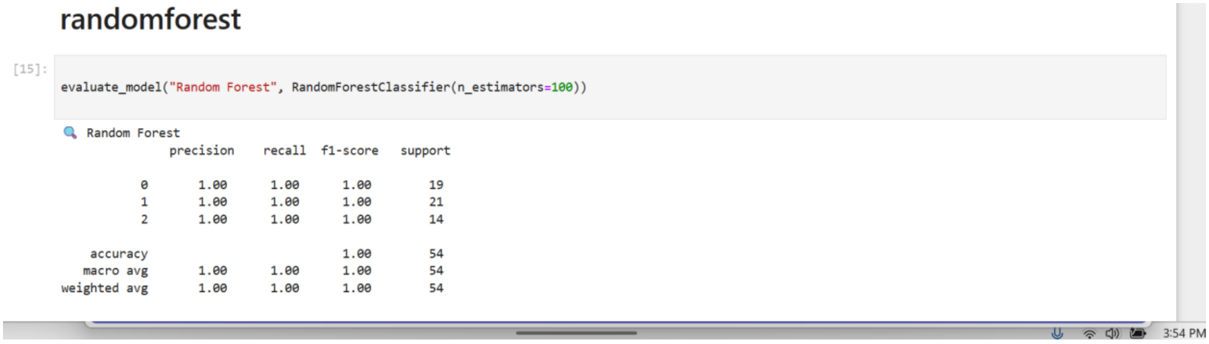
```
[14]: evaluate_model("Decision Tree", DecisionTreeClassifier())
```

```
🔍 Decision Tree
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        19
           1       0.88      1.00      0.93        21
           2       1.00      0.93      0.96        14

    accuracy                           0.94        54
   macro avg       0.96      0.94      0.95        54
weighted avg       0.95      0.94      0.94        54
```



Confusion Matrix: Decision Tree

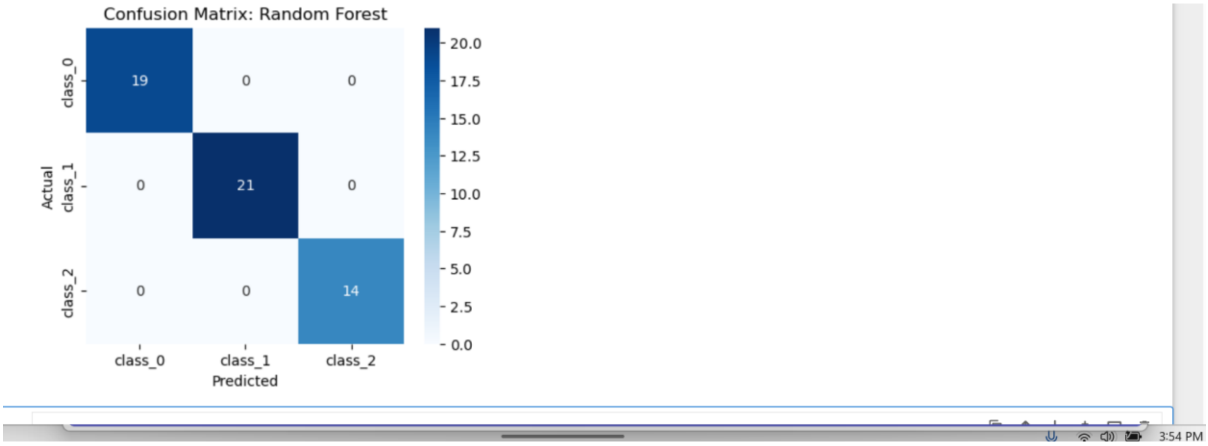
Actual \ Predicted	class_0	class_1	class_2
class_0	17	2	0
class_1	0	21	0
class_2	0	1	13



randomforest

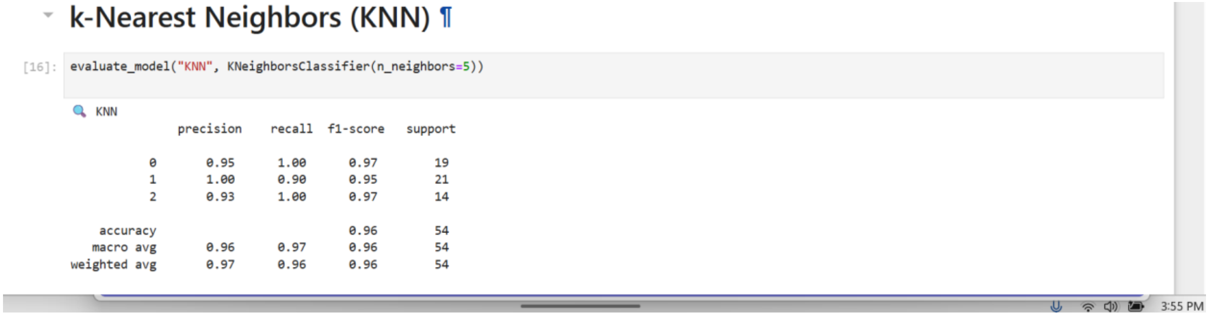
```
[15]:
evaluate_model("Random Forest", RandomForestClassifier(n_estimators=100))
```

```
🔍 Random Forest
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        21
           2       1.00      1.00      1.00        14

    accuracy                           1.00        54
   macro avg       1.00      1.00      1.00        54
weighted avg       1.00      1.00      1.00        54
```



Confusion Matrix: Random Forest

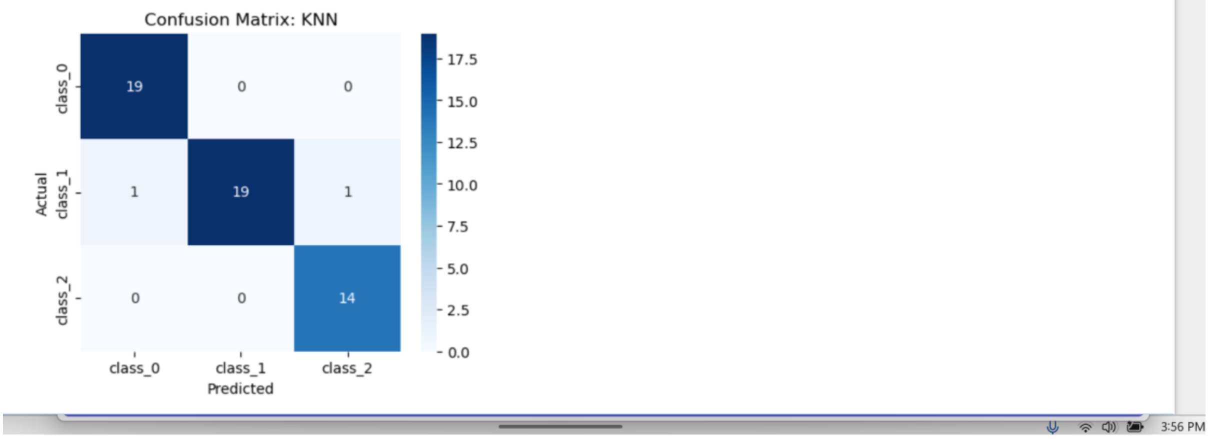
Actual \ Predicted	class_0	class_1	class_2
class_0	19	0	0
class_1	0	21	0
class_2	0	0	14

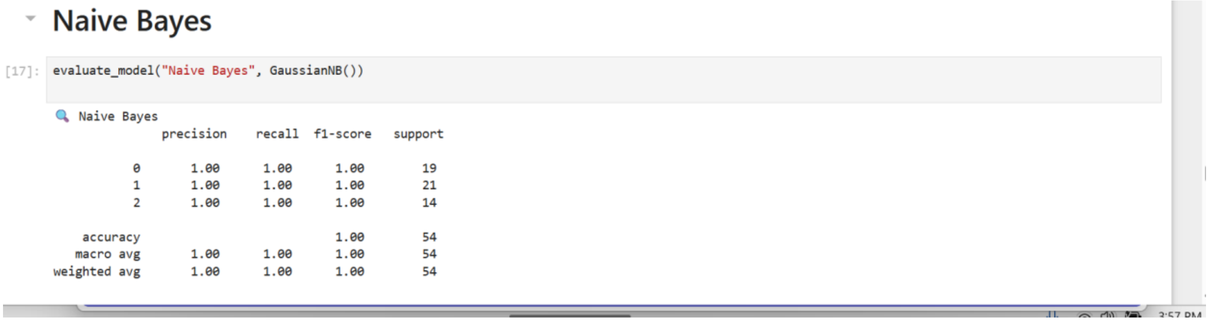


k-Nearest Neighbors (KNN) ¶

```
[16]: evaluate_model("KNN", KNeighborsClassifier(n_neighbors=5))
```

```
🔍 KNN
              precision    recall  f1-score   support

           0       0.95      1.00      0.97        19
           1       1.00      0.90      0.95        21
           2       0.93      1.00      0.97        14

    accuracy                           0.96        54
   macro avg       0.96      0.97      0.96        54
weighted avg       0.97      0.96      0.96        54
```

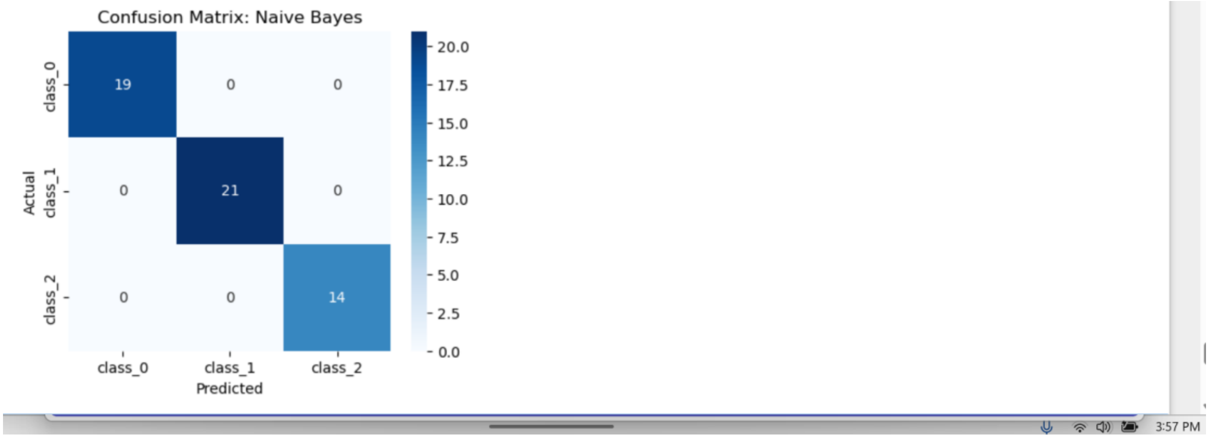


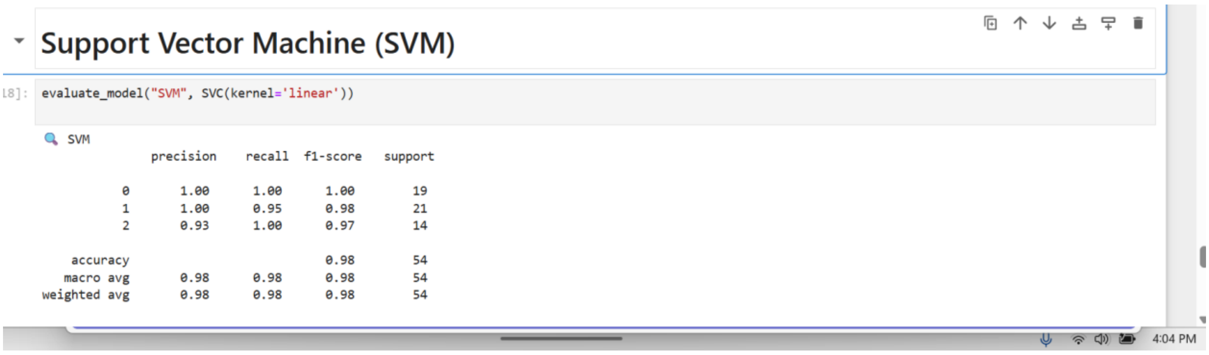


Naive Bayes

```
[17]: evaluate_model("Naive Bayes", GaussianNB())
```

```
🔍 Naive Bayes
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        21
           2       1.00      1.00      1.00        14

    accuracy                           1.00        54
   macro avg       1.00      1.00      1.00        54
weighted avg       1.00      1.00      1.00        54
```

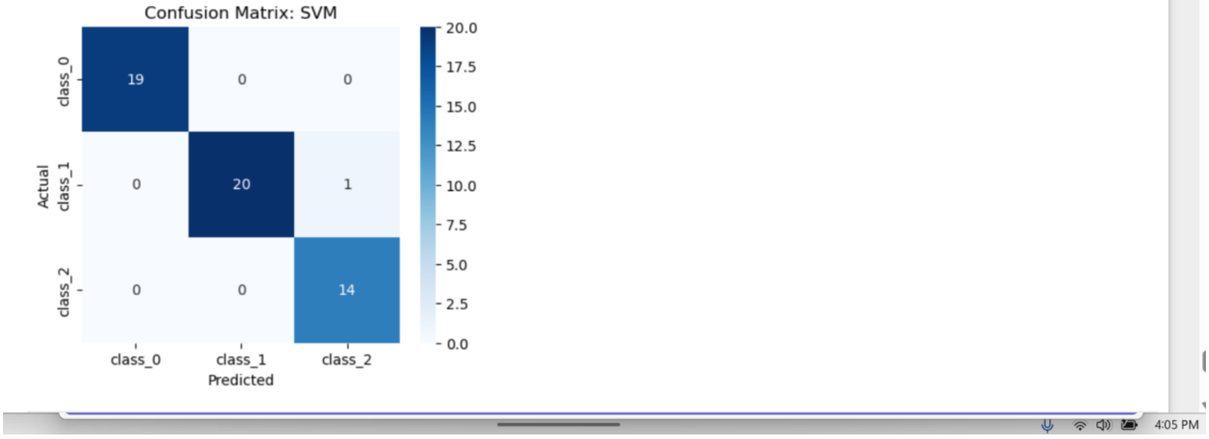




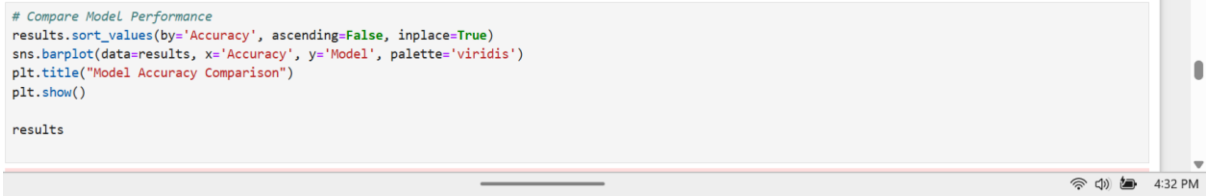
Support Vector Machine (SVM)

```
[18]: evaluate_model("SVM", SVC(kernel='linear'))
```

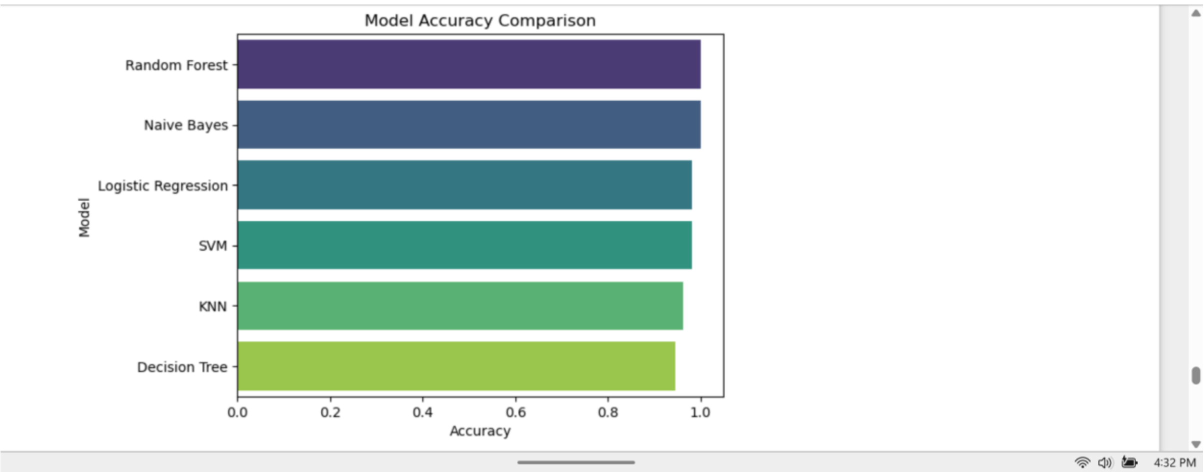
```
🔍 SVM
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.95      0.98        21
           2       0.93      1.00      0.97        14

    accuracy                           0.98        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54
```

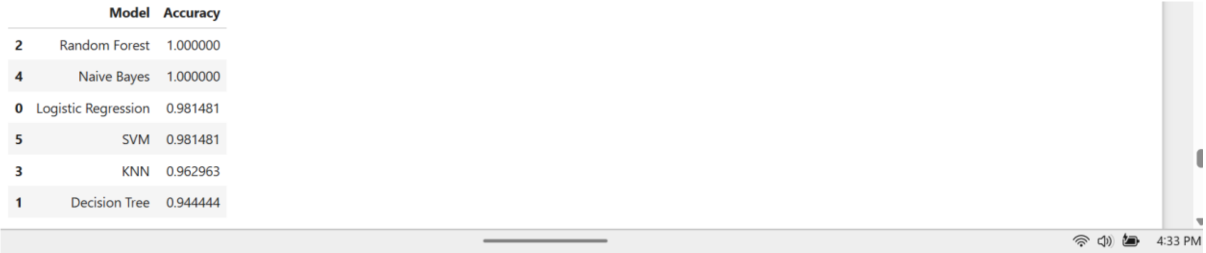


Summary of Model Performance (Accuracy):



```
# Compare Model Performance
results.sort_values(by='Accuracy', ascending=False, inplace=True)
sns.barplot(data=results, x='Accuracy', y='Model', palette='viridis')
plt.title("Model Accuracy Comparison")
plt.show()

results
```





	Model	Accuracy
2	Random Forest	1.000000
4	Naive Bayes	1.000000
0	Logistic Regression	0.981481
5	SVM	0.981481
3	KNN	0.962963
1	Decision Tree	0.944444

5. Model Comparison and Insights

- After evaluating all models, Random Forest achieved the highest accuracy with strong precision and recall across all classes. It performed better due to its ensemble nature, reducing overfitting and capturing complex patterns.
- Logistic Regression and SVM also performed well.
- Naive Bayes, although simple, was surprisingly effective.
- Decision Tree and KNN had decent performance but showed more variance.

Based on these results, Random Forest is the most reliable choice for this dataset.

Link to Code:

https://github.com/shirleensimon/classification-models/blob/main/classification_models.ipynb

Conclusion

This assignment successfully demonstrated the process of applying multiple supervised classification algorithms to a real-world dataset. Naive Bayes and Random Forest emerged as the top performers in terms of accuracy and evaluation metrics. However, each model has its strengths and use cases.