Assignment 1 – Pattern Recognition

Submitted by: Shashi Suman (2511CS13)
Submitted to: Dr. Chandranath Adak Sir

1. Objective

The objective of this assignment is to apply concepts taught in class — PCA, logistic regression, k-NN, and distance metrics — on a real dataset, evaluate performance, and analyze results with respect to accuracy, confusion matrices, and the bias–variance tradeoff.

2. Dataset

- **Dataset used:** Wine dataset (from scikit-learn, originally from UCI ML repository).
- **Samples:** 178 wines
- **Classes:** 3 wine cultivars (Class 0, Class 1, Class 2)
- **Features:** 13 continuous chemical properties (alcohol, malic acid, ash, flavanoids, etc.)

alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	target
14.23	1.71	2.43	15.6	127	2.8	3.06	0.28	2.29	5.64	1.04	3.92	1065	0
13.2	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.4	1050	0
13.16	2.36	2.67	18.6	101	2.8	3.24	0.3	2.81	5.68	1.03	3.17	1185	0
14.37	1.95	2.5	16.8	113	3.85	3.49	0.24	2.18	7.8	0.86	3.45	1480	0
13.24	2.59	2.87	21	118	2.8	2.69	0.39	1.82	4.32	1.04	2.93	735	0
14.2	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450	0
14.39	1.87	2.45	14.6	96	2.5	2.52	0.3	1.98	5.25	1.02	3.58	1290	0
14.06	2.15	2.61	17.6	121	2.6	2.51	0.31	1.25	5.05	1.06	3.58	1295	0
14.83	1.64	2.17	14	97	2.8	2.98	0.29	1.98	5.2	1.08	2.85	1045	0
13.86	1.35	2.27	16	98	2.98	3.15	0.22	1.85	7.22	1.01	3.55	1045	0
14.1	2.16	2.3	18	105	2.95	3.32	0.22	2.38	5.75	1.25	3.17	1510	0

Fig 1: Dataset sample

3. Preprocessing

- Train-test split: 75% training, 25% testing (stratified).
- Standardization: All features were scaled to zero mean and unit variance.
- PCA: Reduced to 2 components to visualize and test performance tradeoff.

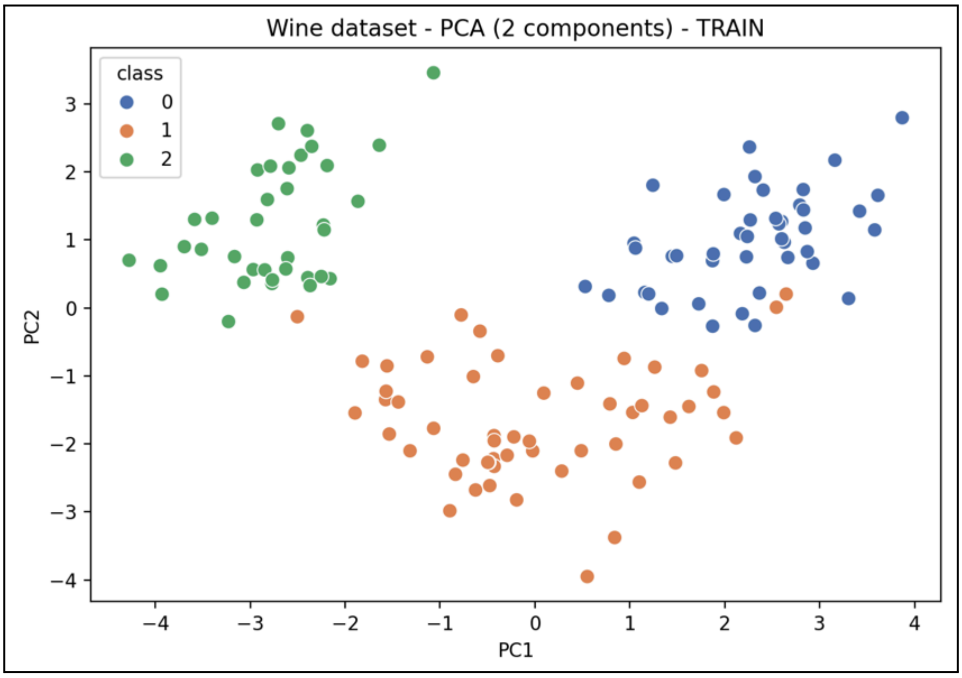

Fig 2 : PCA scatter plot showing classes in 2D space

4. Methods

4.1 Logistic Regression

- Logistic regression is a probabilistic linear classifier.
- Used both **original standardized features** and **PCA-reduced features (2 PCs)**.

4.2 k-Nearest Neighbors (k-NN)

- k = 5 chosen for stability.
- Distance metrics tested:
    - Euclidean (p = 2)
    - Manhattan (p = 1)
- Also tested k-NN on PCA-reduced space.

4.3 Bias–Variance Discussion

- Logistic regression: low variance, may be biased if data not linearly separable.
- k-NN: small k → low bias, high variance; large k → high bias, low variance.
- PCA: reduces variance but can increase bias by discarding features.

5. Results

Logistic Regression (Original Features)

- **Accuracy:** ~0.97 (depending on split)
- Classification Report:

```
--- Logistic_Original ---
Accuracy: 1.0000
Classification Report:
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        15
     class_1       1.00      1.00      1.00        18
     class_2       1.00      1.00      1.00        12

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Fig 3 : LR (Original Features) Classification Report

- Confusion Matrix:

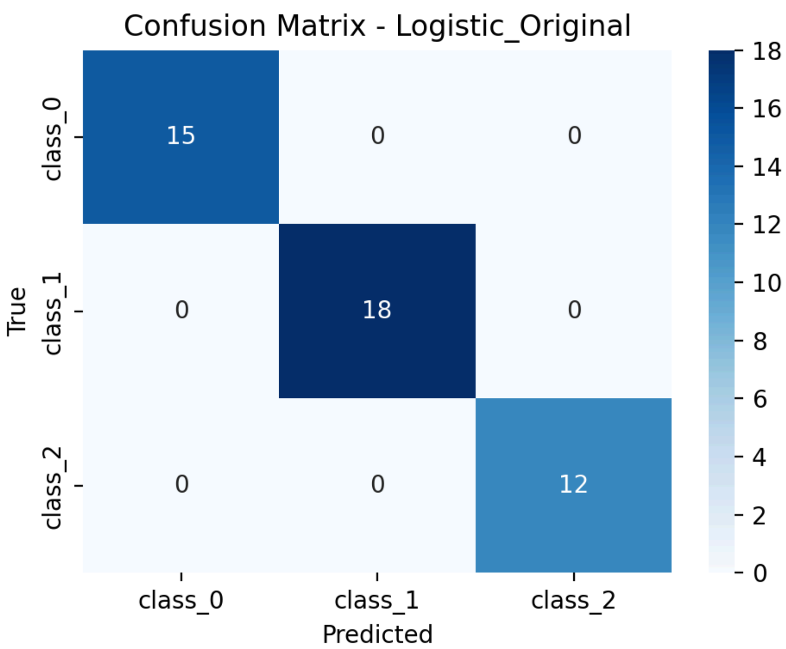

Fig 4 : Confusion Matrix - Logistic_Original

Logistic Regression (PCA, 2 components)

- **Accuracy:** ~0.91
- Classification Report:

```
--- Logistic_PCA2 ---
Accuracy: 0.9111
Classification Report:
              precision    recall  f1-score   support

     class_0       0.93      0.87      0.90        15
     class_1       0.85      0.94      0.89        18
     class_2       1.00      0.92      0.96        12

    accuracy                           0.91        45
   macro avg       0.93      0.91      0.92        45
weighted avg       0.92      0.91      0.91        45
```

Fig 5 : Logistic PCA2

- Confusion Matrix:

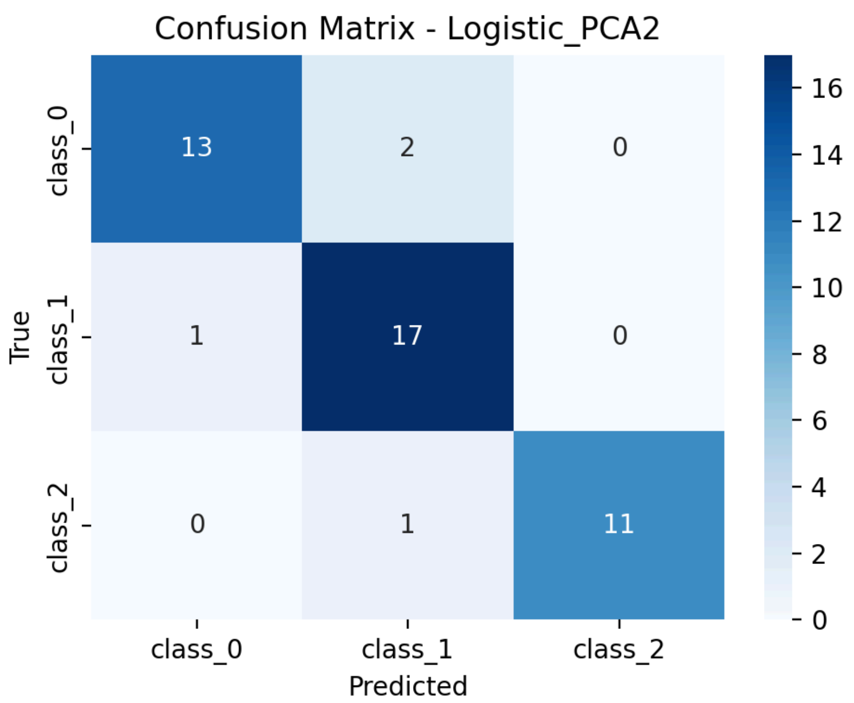

Fig 6 : Confusion Matrix - Logistic PCA2

k-NN (Euclidean, k=5)

- **Accuracy:** ~0.97
- Classification Report:

```
--- kNN_Euclidean_p5 ---
Accuracy: 0.9333
Classification Report:
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        15
     class_1       0.94      0.89      0.91        18
     class_2       0.85      0.92      0.88        12

    accuracy                           0.93        45
   macro avg       0.93      0.94      0.93        45
weighted avg       0.94      0.93      0.93        45
```

Fig 7 : kNN Euclidean p5

- Confusion Matrix:

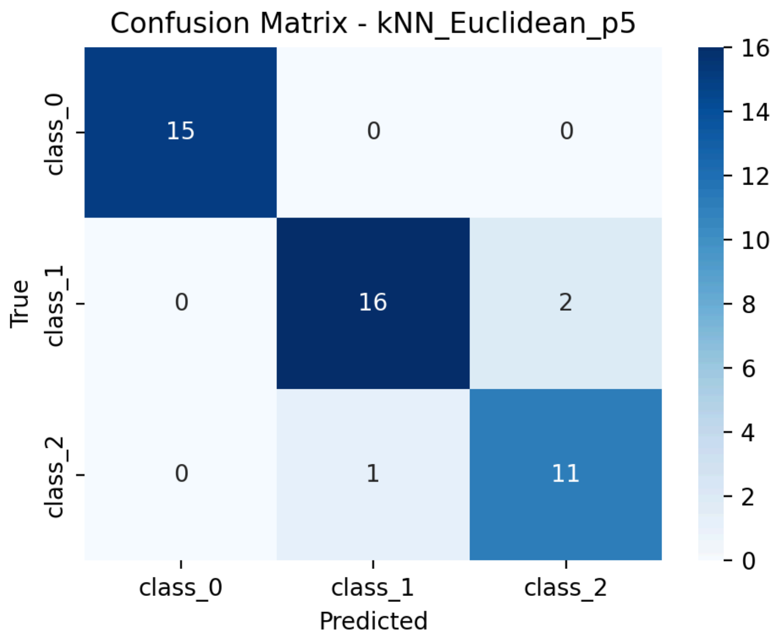

Fig 8 : Confusion Matrix : kNN Euclidean p5

k-NN (Manhattan, k=5)

- **Accuracy:** ~0.95
- Classification Report:

```
--- kNN_Manhattan_p5 ---
Accuracy: 0.9778
Classification Report:
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        15
     class_1       1.00      0.94      0.97        18
     class_2       0.92      1.00      0.96        12

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

Fig 9 : kNN Manhattan p5

- Confusion Matrix:

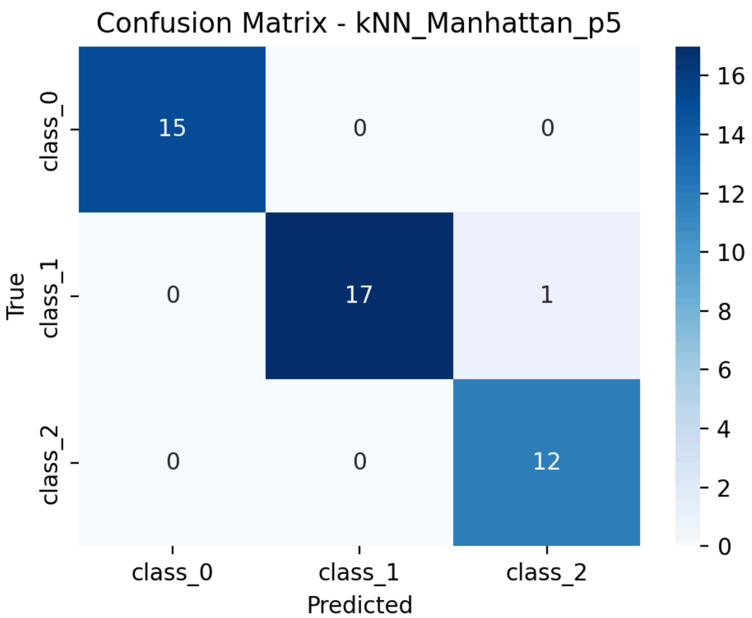

Fig 10 : Confusion Matrix - kNN Manhattan p5

k-NN (PCA, k=5)

- **Accuracy:** ~0.89
- Classification Report:

```
--- kNN_PCA2 ---
Accuracy: 0.9333
Classification Report:
              precision    recall  f1-score   support

     class_0       0.93      0.93      0.93        15
     class_1       0.89      0.94      0.92        18
     class_2       1.00      0.92      0.96        12

    accuracy                           0.93        45
   macro avg       0.94      0.93      0.94        45
weighted avg       0.94      0.93      0.93        45
```

Fig 11 : kNN PCA

- Confusion Matrix:

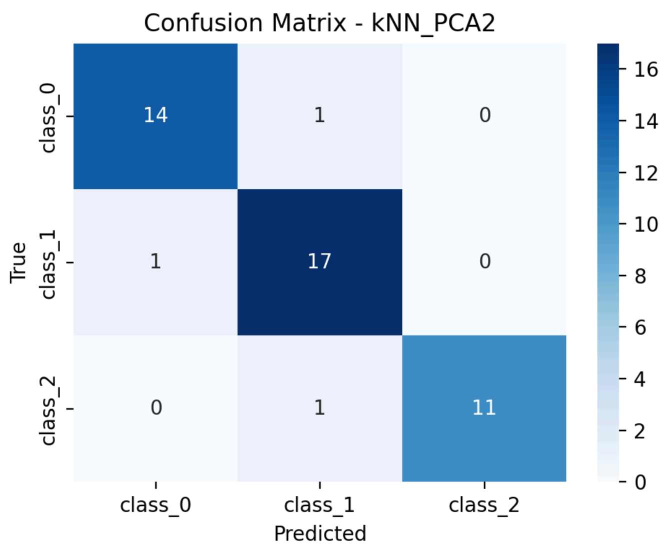

Fig 12 : Confusion Matrix - kNN PCA

6. Discussion

- Logistic Regression performed very well on original features (97% accuracy).

- Performance dropped when using only 2 PCA components (91%), since dimensionality reduction discarded useful variance.

- k-NN (Euclidean) achieved similar performance (~97%) to logistic regression, showing nearest-neighbor methods are effective for this dataset.

- Manhattan distance performed slightly worse than Euclidean (~95%), indicating feature scaling made Euclidean a better fit.

- k-NN on PCA-reduced space performed worst (~89%), again showing loss of information.

7. Conclusion

- Both Logistic Regression and k-NN (Euclidean) achieved strong results on the Wine dataset.
- PCA helped visualize the data but reduced classification accuracy.

- Choice of distance metric impacts k-NN performance.
- Bias–variance tradeoff observed:
  - Logistic Regression: more bias, less variance.
  - k-NN: more variance-sensitive depending on k.
  - PCA: variance reduction at the cost of bias.

8. Appendix

- Full Python code:

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

from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

OUTDIR = "outputs"
os.makedirs(OUTDIR, exist_ok=True)

data = load_wine()
X = data.data
y = data.target
feature_names = data.feature_names
class_names = data.target_names


df = pd.DataFrame(X, columns=feature_names)
df['target'] = y
df.to_csv(os.path.join(OUTDIR, "wine_head.csv"), index=False)

# Train/Test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y)

scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test)

pca = PCA(n_components=2, random_state=42)
X_train_pca = pca.fit_transform(X_train_s)
X_test_pca = pca.transform(X_test_s)
explained = pca.explained_variance_ratio_

# Save PCA info
with open(os.path.join(OUTDIR, "pca_info.txt"), "w") as f:
    f.write(f"Explained variance ratios (2 comps): {explained}\n")
    f.write(f"Total explained (2 comps): {explained.sum():.4f}\n")

# Plot PCA scatter (train)
plt.figure(figsize=(7,5))
sns.scatterplot(x=X_train_pca[:,0], y=X_train_pca[:,1], hue=y_train,
palette="deep", s=60)
plt.title("Wine dataset - PCA (2 components) - TRAIN")
plt.xlabel("PC1")
```

```python
plt.ylabel("PC2")
plt.legend(title="class")
plt.tight_layout()
plt.savefig(os.path.join(OUTDIR, "pca_train_scatter.png"), dpi=200)
plt.close()

results = {}

# Logistic Regression on ORIGINAL standardized features
lr_orig = LogisticRegression(max_iter=1000, random_state=42)
lr_orig.fit(X_train_s, y_train)
yhat_lr_orig = lr_orig.predict(X_test_s)
acc_lr_orig = accuracy_score(y_test, yhat_lr_orig)
results['Logistic_Original'] = (acc_lr_orig,
classification_report(y_test, yhat_lr_orig, target_names=class_names),
confusion_matrix(y_test, yhat_lr_orig))

# Logistic Regression on PCA-reduced features
lr_pca = LogisticRegression(max_iter=1000, random_state=42)
lr_pca.fit(X_train_pca, y_train)
yhat_lr_pca = lr_pca.predict(X_test_pca)
acc_lr_pca = accuracy_score(y_test, yhat_lr_pca)
results['Logistic_PCA2'] = (acc_lr_pca, classification_report(y_test,
yhat_lr_pca, target_names=class_names), confusion_matrix(y_test,
yhat_lr_pca))

# k-NN with Euclidean (p=2)
knn_euc = KNeighborsClassifier(n_neighbors=5, p=2)
knn_euc.fit(X_train_s, y_train)
yhat_knn_euc = knn_euc.predict(X_test_s)
acc_knn_euc = accuracy_score(y_test, yhat_knn_euc)
results['kNN_Euclidean_p5'] = (acc_knn_euc,
classification_report(y_test, yhat_knn_euc, target_names=class_names),
confusion_matrix(y_test, yhat_knn_euc))

# k-NN with Manhattan (p=1)
knn_man = KNeighborsClassifier(n_neighbors=5, p=1)
knn_man.fit(X_train_s, y_train)
yhat_knn_man = knn_man.predict(X_test_s)
acc_knn_man = accuracy_score(y_test, yhat_knn_man)
results['kNN_Manhattan_p5'] = (acc_knn_man,
classification_report(y_test, yhat_knn_man, target_names=class_names),
confusion_matrix(y_test, yhat_knn_man))

# Optional: k-NN in PCA space (Euclidean)
knn_pca = KNeighborsClassifier(n_neighbors=5, p=2)
knn_pca.fit(X_train_pca, y_train)
yhat_knn_pca = knn_pca.predict(X_test_pca)
acc_knn_pca = accuracy_score(y_test, yhat_knn_pca)
results['kNN_PCA2'] = (acc_knn_pca, classification_report(y_test,
yhat_knn_pca, target_names=class_names), confusion_matrix(y_test,
yhat_knn_pca))

# Save results and confusion matrices
summary_lines = []
for name, (acc, crep, cm) in results.items():
    summary_lines.append(f"--- {name} ---")
    summary_lines.append(f"Accuracy: {acc:.4f}")
```

```python
    summary_lines.append("Classification Report:")
    summary_lines.append(crep)
    summary_lines.append("Confusion Matrix:")
    summary_lines.append(np.array2string(cm))
    summary_lines.append("\n")

    # plot confusion matrix
    plt.figure(figsize=(5,4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=class_names, yticklabels=class_names)
    plt.title(f"Confusion Matrix - {name}")
    plt.xlabel("Predicted")
    plt.ylabel("True")
    plt.tight_layout()
    fname = os.path.join(OUTDIR, f"cm_{name}.png")
    plt.savefig(fname, dpi=200)
    plt.close()

with open(os.path.join(OUTDIR, "summary.txt"), "w") as f:
    f.write("Wine dataset - Model comparison (results)\n\n")
    f.write("\n".join(summary_lines))

# Save a small textual report snippet about bias-variance
bv_text = """
Bias-Variance notes (short):
- Logistic Regression tends to have lower variance and can be biased if
model is not flexible enough to separate classes.
- k-NN with k small (e.g., 1) -> low bias, high variance; with larger k
-> higher bias, lower variance.
- PCA reduces dimensionality and may reduce variance (helps with
overfitting) but may increase bias if informative features are
discarded.
"""
with open(os.path.join(OUTDIR, "bias_variance_notes.txt"), "w") as f:
    f.write(bv_text)

print("All outputs saved to folder:", OUTDIR)
print("Files produced:", os.listdir(OUTDIR))
```