**Machine Learning- Practical File**

**Shashwat Sharma (22058570029)**

**BSc (H) Computer Science**

**Making prediction models for the following:**

**1. Naïve Bayes Classifier**

**2. Simple Linear Regression**

**3. Multiple linear regression**

**4. Polynomial Regression**

**5. Lasso and Ridge Regression**

**6. Logistic regression**

**7. Artificial Neural Network**

**8. K-NN classifier**

**9. Decision tree classification**

**10. SVM classification**

**11. K-means clustering**

**12. Hierarchical clustering**

**For evaluation of the regression/classification models, performing following experiments:**

**● Split datasets into training and test sets and evaluate the decision models**

**● Perform k-cross-validation on datasets for evaluation**

**Reporting the efficacy of the machine learning models as follows:**

**● MSE and R2 score for regression models**

**● Accuracy, TP, TN, FP, TN, error, Recall, Specificity, F1-score, AUC for**

**classification models**

**1. Naïve Bayes Classification**

import numpy as np

import pandas as pd

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_auc\_score

from sklearn.preprocessing import StandardScaler

from sklearn.naive\_bayes import GaussianNB

data = load\_breast\_cancer()

X, y = data.data, data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = GaussianNB()

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

y\_pred = model.predict(X\_test)

y\_pred\_proba = model.predict\_proba(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

tn, fp, fn, tp = cm.ravel()

report = classification\_report(y\_test, y\_pred, output\_dict=True)

auc = roc\_auc\_score(y\_test, y\_pred\_proba[:, 1])

print(f”Naïve Bayes Classifier")

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.4f}")

print(f"TP: {tp}, TN: {tn}, FP: {fp}, FN: {fn}")

print(f"Recall: {report['1']['recall']:.4f}")

print(f"F1-score: {report['1']['f1-score']:.4f}")

print(f"AUC: {auc:.4f}")

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_acc = cross\_val\_score(model, X, y, cv=kf, scoring='accuracy').mean()

print(f"Cross-validated Accuracy (5-fold): {cv\_acc:.4f}")

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**2. Simple Linear Regression**

from sklearn.datasets import fetch\_california\_housing

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.model\_selection import train\_test\_split, KFold, cross\_val\_score

data = fetch\_california\_housing()

X, y = data.data, data.target

X = X[:, [0]]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = LinearRegression()

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

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("Simple Linear Regression")

print(f"Feature used: {data.feature\_names[0]}")

print(f"MSE: {mse:.4f}")

print(f"R² Score: {r2:.4f}")

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_scores = cross\_val\_score(model, X, y, cv=kf, scoring='r2')

print(f"Cross-validated R² (5-fold): {cv\_scores.mean():.4f}")

**Output:**

**A black screen with white text

AI-generated content may be incorrect.**

**3. Multiple Linear Regression**

from sklearn.datasets import fetch\_california\_housing

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.model\_selection import train\_test\_split, KFold, cross\_val\_score

data = fetch\_california\_housing()

X, y = data.data, data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = LinearRegression()

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

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("Multiple Linear Regression")

print(f"Feature used: {data.feature\_names}")

print(f"MSE: {mse:.4f}")

print(f"R² Score: {r2:.4f}")

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_scores = cross\_val\_score(model, X, y, cv=kf, scoring='r2')

print(f"Cross-validated R² (5-fold): {cv\_scores.mean():.4f}")

**Output:**

**A black screen with white text

AI-generated content may be incorrect.**

**4. Polynomial Regression**

from sklearn.datasets import fetch\_california\_housing

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.model\_selection import train\_test\_split, KFold, cross\_val\_score

from sklearn.pipeline import make\_pipeline

data = fetch\_california\_housing()

X, y = data.data[:, [0, 2]], data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = make\_pipeline(PolynomialFeatures(degree=3), LinearRegression())

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

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("Polynomial Regression")

print(f"Feature used: {data.feature\_names[0], data.feature\_names[2]}")

print(f"MSE: {mse:.4f}")

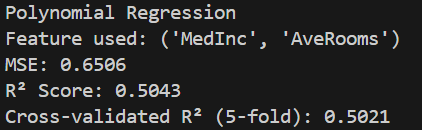
print(f"R² Score: {r2:.4f}")

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_scores = cross\_val\_score(model, X, y, cv=kf, scoring='r2')

print(f"Cross-validated R² (5-fold): {cv\_scores.mean():.4f}")

**Output:**

****

**5. Lasso and Ridge Regression**

from sklearn.datasets import fetch\_california\_housing

from sklearn.linear\_model import Lasso, Ridge

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.model\_selection import train\_test\_split, KFold, cross\_val\_score

from sklearn.pipeline import make\_pipeline

data = fetch\_california\_housing()

X, y = data.data[:, [0, 2]], data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

lasso\_model = make\_pipeline(StandardScaler(), Lasso(alpha=0.1))

lasso\_model.fit(X\_train, y\_train)

y\_pred\_lasso = lasso\_model.predict(X\_test)

ridge\_model = make\_pipeline(StandardScaler(), Ridge(alpha=1.0))

ridge\_model.fit(X\_train, y\_train)

y\_pred\_ridge = ridge\_model.predict(X\_test)

print("Lasso Regression")

print(f"MSE: {mean\_squared\_error(y\_test, y\_pred\_lasso):.4f}")

print(f"R² Score: {r2\_score(y\_test, y\_pred\_lasso):.4f}")

print("\nRidge Regression")

print(f"MSE: {mean\_squared\_error(y\_test, y\_pred\_ridge):.4f}")

print(f"R² Score: {r2\_score(y\_test, y\_pred\_ridge):.4f}")

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

lasso\_cv = cross\_val\_score(lasso\_model, X, y, cv=kf, scoring='r2')

ridge\_cv = cross\_val\_score(ridge\_model, X, y, cv=kf, scoring='r2')

print(f"\nLasso Cross-validated R² (5-fold): {lasso\_cv.mean():.4f}")

print(f"Ridge Cross-validated R² (5-fold): {ridge\_cv.mean():.4f}")

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**6. Logistic Regression**

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_auc\_score

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

data = load\_breast\_cancer()

X, y = data.data, data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = LogisticRegression(max\_iter=1000, solver='liblinear')

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

y\_pred = model.predict(X\_test)

y\_pred\_proba = model.predict\_proba(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

tn, fp, fn, tp = cm.ravel()

report = classification\_report(y\_test, y\_pred, output\_dict=True)

auc = roc\_auc\_score(y\_test, y\_pred\_proba[:, 1])

print(f"Logistic Regression")

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.4f}")

print(f"TP: {tp}, TN: {tn}, FP: {fp}, FN: {fn}")

print(f"Recall: {report['1']['recall']:.4f}")

print(f"F1-score: {report['1']['f1-score']:.4f}")

print(f"AUC: {auc:.4f}")

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_acc = cross\_val\_score(model, X, y, cv=kf, scoring='accuracy').mean()

print(f"Cross-validated Accuracy (5-fold): {cv\_acc:.4f}")

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**7. Artificial Neural Network**

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_auc\_score

from sklearn.preprocessing import StandardScaler

from sklearn.neural\_network import MLPClassifier

data = load\_breast\_cancer()

X, y = data.data, data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = MLPClassifier(hidden\_layer\_sizes=(30,), max\_iter=1000, random\_state=42)

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

y\_pred = model.predict(X\_test)

y\_pred\_proba = model.predict\_proba(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

tn, fp, fn, tp = cm.ravel()

report = classification\_report(y\_test, y\_pred, output\_dict=True)

auc = roc\_auc\_score(y\_test, y\_pred\_proba[:, 1])

print(f"Artificial Neural Network")

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.4f}")

print(f"TP: {tp}, TN: {tn}, FP: {fp}, FN: {fn}")

print(f"Recall: {report['1']['recall']:.4f}")

print(f"F1-score: {report['1']['f1-score']:.4f}")

print(f"AUC: {auc:.4f}")

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_acc = cross\_val\_score(model, X, y, cv=kf, scoring='accuracy').mean()

print(f"Cross-validated Accuracy (5-fold): {cv\_acc:.4f}")

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**8. KNN Classifier**

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_auc\_score

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

data = load\_breast\_cancer()

X, y = data.data, data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = KNeighborsClassifier(n\_neighbors=5)

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

y\_pred = model.predict(X\_test)

y\_pred\_proba = model.predict\_proba(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

tn, fp, fn, tp = cm.ravel()

report = classification\_report(y\_test, y\_pred, output\_dict=True)

auc = roc\_auc\_score(y\_test, y\_pred\_proba[:, 1])

print(f"K Nearest Neighbours")

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.4f}")

print(f"TP: {tp}, TN: {tn}, FP: {fp}, FN: {fn}")

print(f"Recall: {report['1']['recall']:.4f}")

print(f"F1-score: {report['1']['f1-score']:.4f}")

print(f"AUC: {auc:.4f}")

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_acc = cross\_val\_score(model, X, y, cv=kf, scoring='accuracy').mean()

print(f"Cross-validated Accuracy (5-fold): {cv\_acc:.4f}")

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**9. Decision Tree Classifier**

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_auc\_score

from sklearn.preprocessing import StandardScaler

from sklearn.tree import DecisionTreeClassifier

data = load\_breast\_cancer()

X, y = data.data, data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = DecisionTreeClassifier(random\_state=42)

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

y\_pred = model.predict(X\_test)

y\_pred\_proba = model.predict\_proba(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

tn, fp, fn, tp = cm.ravel()

report = classification\_report(y\_test, y\_pred, output\_dict=True)

auc = roc\_auc\_score(y\_test, y\_pred\_proba[:, 1])

print(f"Decision Tree Classifier")

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.4f}")

print(f"TP: {tp}, TN: {tn}, FP: {fp}, FN: {fn}")

print(f"Recall: {report['1']['recall']:.4f}")

print(f"F1-score: {report['1']['f1-score']:.4f}")

print(f"AUC: {auc:.4f}")

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_acc = cross\_val\_score(model, X, y, cv=kf, scoring='accuracy').mean()

print(f"Cross-validated Accuracy (5-fold): {cv\_acc:.4f}")

**Output:**

**A black screen with white text

AI-generated content may be incorrect.**

**10. SVM Classifier**

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_auc\_score

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

data = load\_breast\_cancer()

X, y = data.data, data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = SVC(probability=True, random\_state=42)

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

y\_pred = model.predict(X\_test)

y\_pred\_proba = model.predict\_proba(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

tn, fp, fn, tp = cm.ravel()

report = classification\_report(y\_test, y\_pred, output\_dict=True)

auc = roc\_auc\_score(y\_test, y\_pred\_proba[:, 1])

print(f"SVM Classifier")

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.4f}")

print(f"TP: {tp}, TN: {tn}, FP: {fp}, FN: {fn}")

print(f"Recall: {report['1']['recall']:.4f}")

print(f"F1-score: {report['1']['f1-score']:.4f}")

print(f"AUC: {auc:.4f}")

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_acc = cross\_val\_score(model, X, y, cv=kf, scoring='accuracy').mean()

print(f"Cross-validated Accuracy (5-fold): {cv\_acc:.4f}")

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**11. k-means Clustering**

from sklearn.datasets import load\_breast\_cancer

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

data = load\_breast\_cancer()

X = data.data

scaler = StandardScaler()

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

kmeans = KMeans(n\_clusters=2, random\_state=42, n\_init=10)

y\_kmeans = kmeans.fit\_predict(X\_scaled)

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_scaled)

plt.figure(figsize=(8, 5))

plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=y\_kmeans, cmap='viridis', s=50)

plt.title("K-Means Clustering (PCA Visualization)")

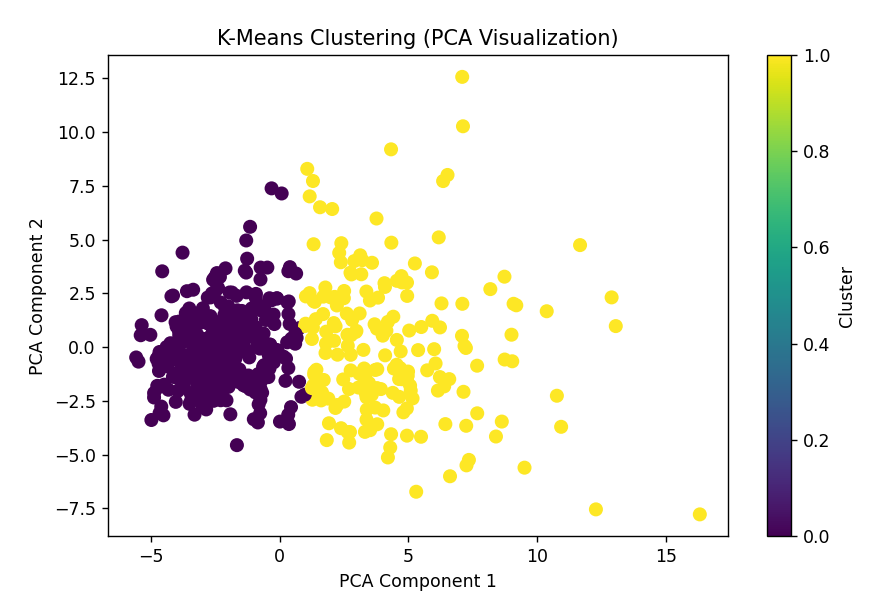
plt.xlabel("PCA Component 1")

plt.ylabel("PCA Component 2")

plt.colorbar(label="Cluster")

plt.show()

**Output:**

****

**12. Hierarchical Clustering**

from sklearn.datasets import load\_breast\_cancer

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import AgglomerativeClustering

from scipy.cluster.hierarchy import dendrogram, linkage

from sklearn.metrics import silhouette\_score, adjusted\_rand\_score

import matplotlib.pyplot as plt

data = load\_breast\_cancer()

X = data.data

scaler = StandardScaler()

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

y\_true = data.target

agglo = AgglomerativeClustering(n\_clusters=2, linkage='single')

y\_pred = agglo.fit\_predict(X\_scaled)

sil\_score = silhouette\_score(X\_scaled, y\_pred)

ari\_score = adjusted\_rand\_score(y\_true, y\_pred)

print("Silhouette Score:", round(sil\_score, 4))

print("Adjusted Rand Index (ARI):", round(ari\_score, 4))

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

linked = linkage(X\_scaled, method='single')

dendrogram(linked, truncate\_mode='level', p=5)

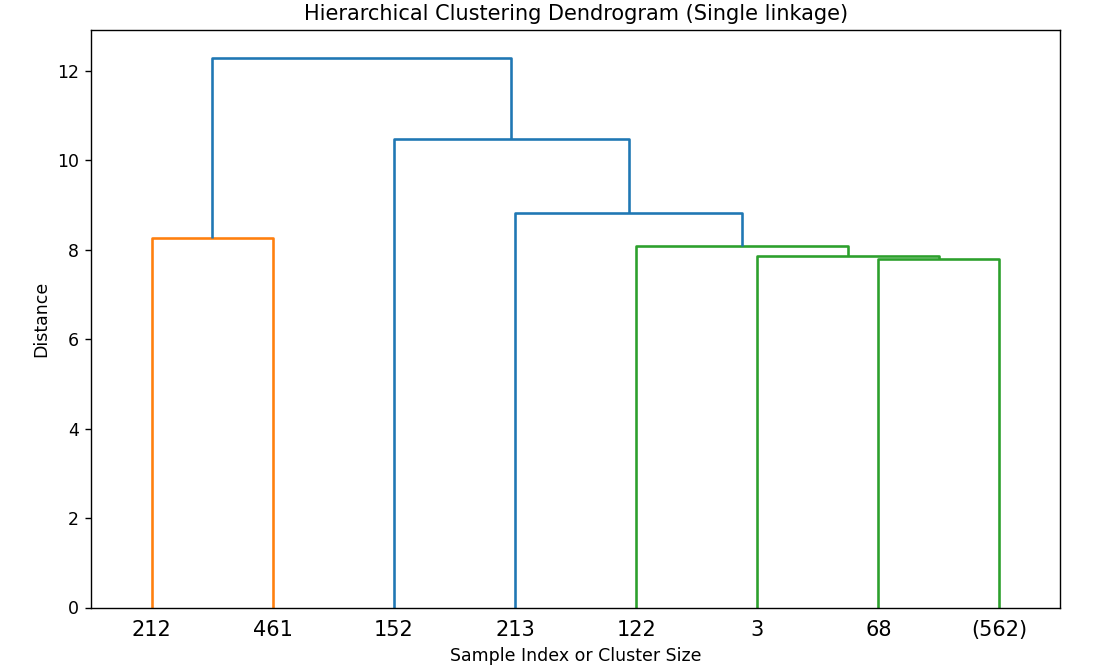
plt.title("Hierarchical Clustering Dendrogram (Single linkage)")

plt.xlabel("Sample Index or Cluster Size")

plt.ylabel("Distance")

plt.show()

**Output:**

****