Rayat Shikshan Sanstha's

R.B. Narayanrao Borawake College, Shrirampur

(Autonomous)

National Education Policy

(Affiliated to Savitribai Phule Pune University)

Two Years Degree Program in Computer Science (Faculty of Science and Technology)

Syllabus under Autonomy and National Education Policy

M. Sc. (Computer Science) Part- II

Practical Assignment Lab Book

Choice Based Credit System [CBCS] Syllabus for National Education Policy to be implemented from Academic Year 2024-2025

Machine Learning (CS-MJ-635P)

Assignment Completion Sheet

Name of the Student:

Roll No:

Sr. No.	Assignment Title	Marks Obtained	Signature of Instructor
1	Write a Python program to prepare a scatter plot for the Iris dataset.		
2	Write a Python program to find all null values in a given dataset and remove them.		
3	Write a Python program to make categorical values in numeric format for a given dataset.		
4	Write a Python program to implement simple linear regression for predicting house price.		
5	Write a Python program to implement multiple linear regression for a given dataset.		
6	Write a Python program to implement polynomial linear regression for a given dataset.		
7	Write a Python program to implement Naive Bayes.		
8	Write a Python program to implement a decision tree to determine whether or not to play tennis.		
9	Write a Python program to implement Linear SVM.		
10	To reduce the 4D data of the Iris flower dataset to 2D using Principal Component Analysis (PCA) and then train a model.		

Total Marks :

Converted into 10 Marks :
Date:

Signature of In Charge **Head**

Internal Examiner **External Examiner**

Assignment No 1

Q.1)Write a Python program to prepare a scatter plot for the Iris dataset.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load Iris dataset
iris = sns.load_dataset('iris')

# Create scatter plot
sns.scatterplot(data=iris, x='sepal_length', y='sepal_width', hue='species')

plt.title('Scatter Plot of Iris Dataset')

plt.show()
```

Assignment Evaluation

0: Not Done		1: Incomplete		2: Late Complete	
3: Need Improvement		4: Completed		5: Well Done	

Date: **Practical In-charge**

Assignment No .2

Q.2)Write a Python program to find all null values in a given dataset and remove them.

```
import pandas as pd

# Load dataset
data = pd.read_csv('your_dataset.csv')

# Find null values
print(data.isnull().sum())

# Remove null values
data_cleaned = data.dropna()


print("\nCleaned Dataset:")

print(cleaned_data)
```

Assignment Evaluation

0: Not Done		1: Incomplete		2: Late Complete	
3: Need Improvement		4: Completed		5: Well Done	

Date: **Practical In-charge**

Assignment No. 3

Q.3) Write a Python program to make categorical values in numeric format for a given dataset.

```
import pandas as pd

# Load the dataset (replace 'your_dataset.csv' with your file path)
file_path = 'your_dataset.csv'
data = pd.read_csv(file_path)

# Display the original dataset
print("Original Dataset:")
print(data)

# Method 1: Label Encoding
label_encoded_data = data.copy()
for column in label_encoded_data.select_dtypes(include=['object']).columns:
label_encoded_data[column] = label_encoded_data[column].astype('category').cat.codes

print("\nLabel Encoded Dataset:")
print(label_encoded_data)

# Method 2: One-Hot Encoding
one_hot_encoded_data = pd.get_dummies(data, drop_first=True)

print("\nOne-Hot Encoded Dataset:")
print(one_hot_encoded_data)

# Optionally, save the encoded datasets to new CSV files

label_encoded_data.to_csv('label_encoded_dataset.csv', index=False)
one_hot_encoded_data.to_csv('one_hot_encoded_dataset.csv', index=False)
print("\nEncoded datasets saved as 'label_encoded_dataset.csv' and
'one_hot_encoded_dataset.csv'")
```

Assignment Evaluation

0: Not Done		1: Incomplete		2: Late Complete	
3: Need Improvement		4: Completed		5: Well Done	

Date: **Practical In-charge**

Assignment No .4

Q.4) Write a Python program to implement simple linear regression for predicting house price.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Load dataset
data = pd.read_csv('house_prices.csv')

# Split dataset
X = data[['feature1', 'feature2']]  # Replace with actual feature names
y = data['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
```

Assignment Evaluation

0: Not Done		1: Incomplete		2: Late Complete	
3: Need Improvement		4: Completed		5: Well Done	

Date: **Practical In-charge**

Assignment No.5

Q.5)Write a Python program to implement multiple linear regression for a given dataset.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Load dataset
data = pd.read_csv('your_dataset.csv')

# Split dataset
X = data[['feature1', 'feature2', 'feature3']]  # Replace with actual features
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
```

Assignment Evaluation

0: Not Done		1: Incomplete		2: Late Complete	
3: Need Improvement		4: Completed		5: Well Done	

Date: **Practical In-charge**

Assignment No.6

Q.6)Write a Python program to implement polynomial linear regression for a given dataset.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# Load dataset
data = pd.read_csv('your_dataset.csv')

# Split dataset
X = data[['feature1']]  # Replace with actual feature
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Polynomial features
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X_train)

# Create and fit model
model = LinearRegression()
model.fit(X_poly, y_train)

# Predictions
X_test_poly = poly.transform(X_test)
predictions = model.predict(X_test_poly)
```

Assignment Evaluation

0: Not Done		1: Incomplet		2: Late Complete	
3: Need Improvement		4: Completed		5: Well Done	

Date: **Practical In-charge**

Assignment No 7

Q.7)Write a Python program to implement Naive Bayes.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

# Load dataset
data = pd.read_csv('your_dataset.csv')

# Split dataset
X = data[['feature1', 'feature2']]  # Replace with actual features
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit model
model = GaussianNB()
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
```

Assignment Evaluation

0: Not Done		1: Incomplete		2: Late Complete	
3: Need Improvement		4: Completed		5: Well Done	

Date: **Practical In-charge**

Assignment No.8

Q.8)Write a Python program to implement a decision tree to determine whether or not to play tennis.

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split

# Load dataset
data = pd.read_csv('tennis.csv')

# Prepare data
X = data[['Outlook', 'Temperature', 'Humidity', 'Wind']]  # Replace with actual features
y = data['Play']

# Convert categorical data to numeric
X = pd.get_dummies(X, drop_first=True)

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
```

Assignment Evaluation

0: Not Done		1: Incomplete		2: Late Complete	
3: Need Improvement		4: Completed		5: Well Done	

Date: **Practical In-charge**

Assignment No.9

Q.9) Write a Python program to implement Linear SVM.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

# Load dataset
data = pd.read_csv('your_dataset.csv')

# Split dataset
X = data[['feature1', 'feature2']]  # Replace with actual features
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit model
model = SVC(kernel='linear')
model.fit(X_train, y_train)

# Predictions
predictions = model.predict(X_test)
```

Assignment Evaluation

0: Not Done		1: Incomplete		2: Late Complete	
3: Need Improvement		4: Completed		5: Well Done	

Date: **Practical In-charge**

Assignment No.10

Q.10) To reduce the 4D data of the Iris flower dataset to 2D using Principal Component Analysis (PCA) and then train a model.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
from sklearn.model_selection  import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Load the Iris dataset
iris = load_iris()
X = iris.data # Features
y = iris.target # Target variable (species)

# Reduce the dimensionality from 4D to 2D using PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Create a DataFrame for the PCA result
df_pca = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'])
df_pca['species'] = y

# Plot the PCA result
plt.figure(figsize=(10, 6))
plt.scatter(df_pca['PC1'], df_pca['PC2'], c=df_pca['species'], cmap='viridis', edgecolor='k',
s=100)
plt.title('PCA of Iris Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Species')
plt.show()

# Split the PCA-transformed data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.2,
random_state=42)
```

```
# Train a KNN model
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = knn_model.predict(X_test)

# Evaluate the model
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Assignment Evaluation

0: Not Done		1: Incomplete		2: Late Complete	
3: Need Improvement		4: Completed		5: Well Done	

Date: **Practical In-charge**