SSN College of Engineering, Chennai Department of Computer Science and Engineering

Date: July 30, 2025
Academic Year: 2025–2026
Degree: M.Tech (5-Year Integrated) CSE
Course Code: ICS1512 Machine Learning Algorithms Laboratory

Experiment 1: Working with Python packages-Numpy, Scipy, Scikit-Learn,Matplotlib

AIM

To explore five machine learning datasets, select and justify the most suitable algorithms for each task, implement a workflow in Python, including data pre-processing, model training, and performance evaluation, and interpret the results using appropriate metrics.

Libraries Used and Key Concepts

This section highlights commonly used Python libraries in machine learning and explores key functions and concepts from each.

1. NumPy

Purpose: Numerical computing and array manipulation.
Key Functions:

- `np.array()`: Create arrays
- `np.mean(), np.std(), np.sum()`: Statistical computations
- `np.dot(), np.linalg.inv()`: Linear algebra

2. Pandas

Purpose: Data manipulation and analysis.
Key Functions:

- `pd.read_csv()`: Load data from CSV
- `df.head()`, `df.describe()`, `df.info()`: Data inspection
- `df.fillna()`, `df.drop()`, `df.groupby()`: Data cleaning and transformation

3. SciPy

Purpose: Advanced scientific computing and mathematical functions.
Key Functions:

- `scipy.stats`: Statistical functions and distributions
- `scipy.optimize`: Optimization routines
- `scipy.spatial`: Distance computations

4. Scikit-learn

Purpose: Machine learning tools for model training, evaluation, and preprocessing.
Key Functions:

- `train_test_split()`, `cross_val_score()`: Data splitting
- `StandardScaler`, `LabelEncoder`: Preprocessing tools
- `LinearRegression`, `DecisionTreeClassifier`, `etc.`: Models
- `classification_report()`, `confusion_matrix()`: Evaluation metrics

5. Matplotlib

Purpose: Visualization and plotting.
Key Functions:

- `plt.plot()`, `plt.scatter()`, `plt.hist()`: Basic plots
- `plt.xlabel()`, `plt.ylabel()`, `plt.title()`: Plot formatting
- `plt.show()`: Display figures

Dataset Analysis, Model Selection, and Performance Metrics

1. Loan Amount Prediction

Type: Supervised Learning
Task: Regression
Dataset: Kaggle Loan Prediction Dataset
Model Rationale: Linear Regression is used for continuous target prediction due to its interpretability and efficiency in modeling linear relationships.

Python Code:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder,
    StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,
    mean_absolute_error, r2_score
import numpy as np

df = pd.read_csv("loan_data.csv")
# For simplicity in this example, we drop rows with any
    missing values
# A more robust approach would be imputation
df.dropna(inplace=True)
df = df.drop('Loan_ID', axis=1) # Drop non-numeric ID

# Convert categorical columns to numeric
```

```python
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])

X = df.drop('LoanAmount', axis=1)
y = df['LoanAmount']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled,
    y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(f"MSE: {mean_squared_error(y_test, y_pred):.2f}")
print(f"MAE: {mean_absolute_error(y_test, y_pred):.2f}")
print(f"R^2: {r2_score(y_test, y_pred):.2f}")

# Example Prediction
sample_input = X_test[0].reshape(1, -1)
sample_prediction = model.predict(sample_input)
actual_value = y_test.iloc[0]
print(f"\nSample Input Features (scaled): {np.round(X_test
    [0], 2)}")
print(f"Predicted Loan Amount: {sample_prediction[0]:.2f}")
print(f"Actual Loan Amount: {actual_value:.2f}")
```

Output and Prediction:

```
MSE: 2235.15
MAE: 35.10
R^2: 0.44

Sample Input Features (scaled): [-0.51  1.03  0.27 -0.28 -0.21  0.58  0.89
    -0.55  0.68  0.28]
Predicted Loan Amount: 147.93
Actual Loan Amount: 155.00
```

2. Handwritten Character Recognition

Type: Supervised Learning
Task: Multiclass Classification
Dataset: MNIST
Model Rationale: CNNs exploit spatial hierarchies in images, automati-

cally learning features like edges and shapes, making them ideal for image classification.

Python Code:

```python
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from keras.datasets import mnist
from keras.utils import to_categorical
from sklearn.metrics import classification_report,
    confusion_matrix
import numpy as np

(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(-1,28,28,1)/255.0
X_test = X_test.reshape(-1,28,28,1)/255.0
y_train_cat = to_categorical(y_train)
y_test_cat = to_categorical(y_test)
model = Sequential([
  Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)),
  MaxPooling2D(), Flatten(),
  Dense(128,activation='relu'), Dense(10,activation='softmax'
    )
])
model.compile('adam','categorical_crossentropy',metrics=['
    accuracy'])
# Using 1 epoch for faster demonstration
model.fit(X_train,y_train_cat,epochs=1,validation_data=(
    X_test,y_test_cat))
y_pred_probs = model.predict(X_test)
y_pred = np.argmax(y_pred_probs, axis=1)
print(classification_report(y_test,y_pred))

# Example Prediction
sample_input = X_test[0].reshape(1, 28, 28, 1)
sample_prediction = np.argmax(model.predict(sample_input),
    axis=1)
print(f"\nPredicted Digit: {sample_prediction[0]}")
print(f"Actual Digit: {y_test[0]}")
```

Output and Prediction:

```
... (Training logs) ...
313/313 [==============================] - 1s 2ms/step
              precision    recall  f1-score   support

```

```
           0       0.99      0.99      0.99       980
           1       0.99      1.00      0.99      1135
           2       0.98      0.98      0.98      1032
           3       0.98      0.98      0.98      1010
           4       0.98      0.98      0.98       982
           5       0.98      0.97      0.98       892
           6       0.99      0.98      0.98       958
           7       0.98      0.98      0.98      1028
           8       0.97      0.97      0.97       974
           9       0.97      0.97      0.97      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000

1/1 [==============================] - 0s 18ms/step
Predicted Digit: 7
Actual Digit: 7
```

3. Email Spam Classification

Type: Supervised Learning
Task: Binary Classification
Dataset: UCI Spambase (using a simplified CSV for demonstration)
Model Rationale: Multinomial Naive Bayes efficiently handles high-dimensional sparse text data by modeling word counts under conditional independence assumptions.

Python Code:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Create a dummy dataframe for demonstration
data = {'text': ['Free money now!', 'Hi mom, how are you?', 'Claim your prize', 'Meeting at 5pm'],
        'label': ['spam', 'ham', 'spam', 'ham']}
df = pd.DataFrame(data)

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df['text'])
y = df['label']
```

```python
X_train,X_test,y_train,y_test = train_test_split(X,y,
    test_size=0.25,random_state=42)
model=MultinomialNB()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test,y_pred):.2f}")
print(classification_report(y_test,y_pred))

# Example Prediction
sample_text = ["click here for free cash"]
sample_vec = vectorizer.transform(sample_text)
prediction = model.predict(sample_vec)
print(f"\nInput Text: '{sample_text[0]}'")
print(f"Predicted Label: {prediction[0]}")
```

Output and Prediction:

```
Accuracy: 1.00
              precision    recall  f1-score   support

         ham       1.00      1.00      1.00         1
   micro avg       1.00      1.00      1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1

Input Text: 'click here for free cash'
Predicted Label: spam
```

4. Predicting Diabetes

Type: Supervised Learning
Task: Binary Classification
Dataset: Pima Indians Diabetes
Model Rationale: Random Forest reduces overfitting by averaging multiple decision trees, captures nonlinear interactions, and provides feature importance for interpretability.

Python Code:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,
    classification_report

```

```
df=pd.read_csv("diabetes.csv")
X=df.drop('Outcome',axis=1)
y=df['Outcome']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size
    =0.2,random_state=42)
model=RandomForestClassifier(random_state=42)
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test,y_pred):.2f}")
print(classification_report(y_test,y_pred))

# Example Prediction
sample_input = X_test.iloc[0].values.reshape(1, -1)
sample_prediction = model.predict(sample_input)
print(f"\nInput Features: {X_test.iloc[0].values}")
print(f"Predicted Outcome (1=Diabetic): {sample_prediction
    [0]}")
print(f"Actual Outcome: {y_test.iloc[0]}")
```

Output and Prediction:

```
Accuracy: 0.72
              precision    recall  f1-score   support

           0       0.80      0.76      0.78        99
           1       0.59      0.65      0.62        55

    accuracy                           0.72       154
   macro avg       0.70      0.71      0.70       154
weighted avg       0.73      0.72      0.72       154

Input Features: [  6.   148.    72.    35.     0.    33.6    0.627  50.  ]
Predicted Outcome (1=Diabetic): 1
Actual Outcome: 1
```

5. Iris Dataset

Type: Supervised Learning
Task: Multiclass Classification
Dataset: UCI Iris
Model Rationale: Decision Trees are chosen for their interpretability, ability to handle multiclass problems, and nonparametric nature for nonlinear decision boundaries.

Python Code:

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

iris=load_iris()
X,y=iris.data,iris.target
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
model=DecisionTreeClassifier()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test,y_pred):.2f}")
print(classification_report(y_test,y_pred,target_names=iris.target_names))

# Example Prediction
sample_input = X_test[0].reshape(1, -1)
sample_prediction_index = model.predict(sample_input)[0]
predicted_species = iris.target_names[sample_prediction_index]
actual_species = iris.target_names[y_test[0]]
print(f"\nInput Features: {X_test[0]}")
print(f"Predicted Species: {predicted_species}")
print(f"Actual Species: {actual_species}")
```

Output and Prediction:

```
Accuracy: 1.00
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

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

Input Features: [6.1 2.8 4.7 1.2]
Predicted Species: versicolor
Actual Species: versicolor
```

Dataset	Type of ML task	Suitable ML Algorithm
Iris Dataset	Multiclass Classification	Decision Tree Classifier
Loan Amount Prediction	Regression	Linear Regression
Predicting Diabetes	Binary Classification	Random Forest Classifier
Email Spam Classification	Binary Classification	Multinomial Naive Bayes
Handwritten Character Recognition	Multiclass Classification	Convolutional Neural Network

Table 1: Overview of Datasets and Selected Algorithms

Summary of Tasks

Learning Outcomes

Learned data loading, preprocessing, selection of a suitable model, implementation, and evaluation for five datasets using five different models. Gained practical experience in interpreting performance metrics such as accuracy, MSE, and classification reports to assess model effectiveness.