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

R Padmashri
3122 23 5001 093
Department of Computer Science and Engineering

January 26, 2026

Abstract

This experiment applies appropriate machine learning algorithms to multiple real-world datasets to understand data preprocessing, feature selection, model training, evaluation, and interpretation of results using Python-based machine learning libraries.

Keywords: Machine Learning, Classification, Feature Selection, Logistic Regression, Naive Bayes, KNN, TensorFlow

1 Aim of the Experiment

The aim of this experiment is to explore and implement different supervised machine learning algorithms on diverse datasets, analyze their performance using standard evaluation metrics, and understand the importance of algorithm selection, preprocessing, and feature selection in real-world machine learning problems.

2 Datasets Used

- Iris Dataset (Multiclass Classification)
- Loan Approval Dataset (Binary Classification)
- Diabetes Prediction Dataset (Binary Classification)
- Email Spam Dataset (Binary Classification)
- Handwritten Digit Dataset (Image Classification)

3 Methodology and Implementation

3.1 Libraries and Dependencies (Imports)

The following Python libraries were used across all experiments for data handling, visualization, machine learning, and deep learning implementation.

Listing 1: Python libraries used in the experiments

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.feature_selection import SelectKBest, chi2, f_classif
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
```

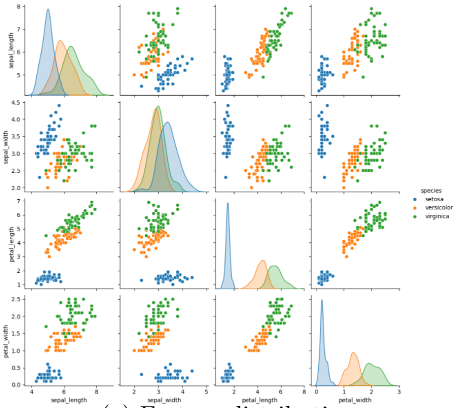
```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

import os
import cv2
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
```

3.2 Iris Dataset using K-Nearest Neighbors

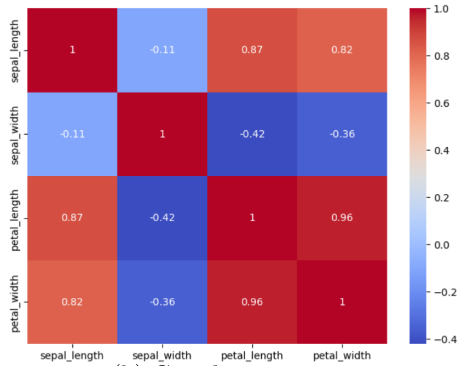
The Iris dataset contains numerical flower measurements used to classify samples into three species. Since the dataset is small and numeric, K-Nearest Neighbors (KNN) was chosen.

Listing 2: EDA and KNN for Iris dataset

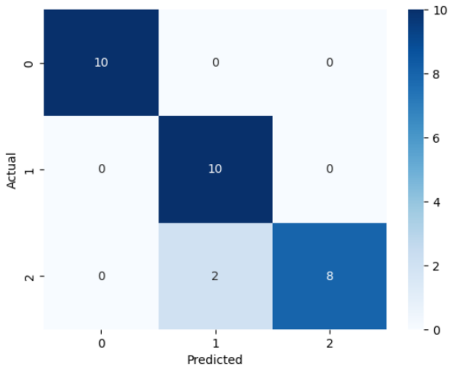
```python
# Basic structure
df.head()
df.describe()

# Feature distribution
df.hist(figsize=(8,6))
plt.show()

# Correlation matrix
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()

# Model and confusion matrix
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

confusion_matrix(y_test, y_pred)
```



(a) Feature distribution



(b) Correlation matrix



(c) Confusion matrix

Figure 1: Exploratory analysis and classification results for the Iris dataset

Table 1: Performance Metrics for Iris Dataset (KNN)

Metric	Value
Accuracy	0.9333
Precision (Macro Avg)	0.93
Recall (Macro Avg)	0.93
F1-Score (Macro Avg)	0.93

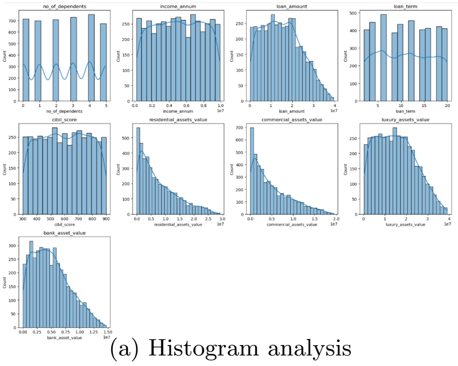
3.3 Loan Approval Prediction using Logistic Regression

Loan approval prediction is a binary classification task. Logistic Regression was used due to its interpretability and suitability for binary outcomes.

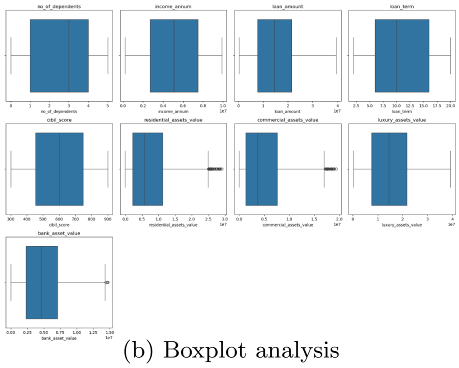
Listing 3: EDA and Logistic Regression for Loan approval prediction

```
# Dataset overview
df.info()
df.describe()

# Histogram analysis
df.hist(figsize=(10,6))
plt.show()

# Boxplot analysis
df.boxplot(figsize=(10,6))
plt.show()

# Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

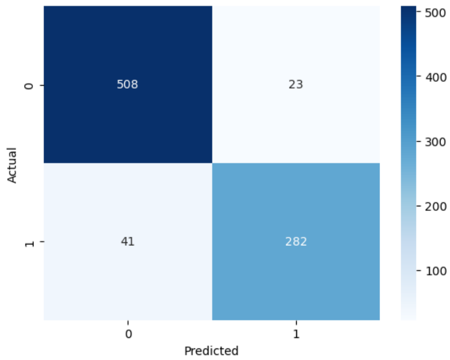
```
y_pred = model.predict(X_test)

confusion_matrix(y_test, y_pred)
```



(a) Histogram analysis



(b) Boxplot analysis



(c) Confusion matrix

Figure 2: Exploratory analysis and results for Loan Approval Prediction

Table 2: Performance Metrics for Loan Approval Prediction

Metric	Value
Accuracy	0.9251
Precision	0.92
Recall	0.91
F1-Score	0.92

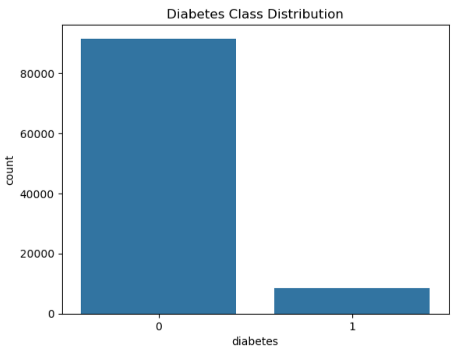
3.4 Diabetes Prediction using Logistic Regression

Medical attributes were used to predict diabetes. Logistic Regression was applied and evaluated using standard performance metrics.

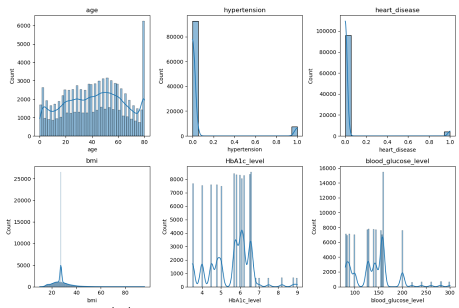
Listing 4: EDA and Logistic Regression for Diabetes dataset

```
# Class distribution
sns.countplot(x=y)
plt.show()

# Feature distribution
df.hist(figsize=(10,6))
plt.show()

# Correlation analysis
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()

```

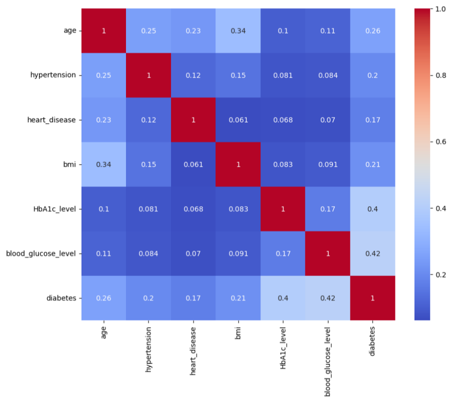
```
# Feature selection
selector = SelectKBest(score_func=f_classif, k=5)
X_selected = selector.fit_transform(X, y)

# Scaling
X_scaled = StandardScaler().fit_transform(X_selected)

# Model training
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

classification_report(y_test, y_pred)
```



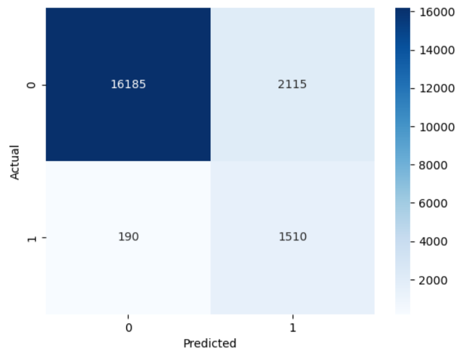
(a) Class distribution



(a) Feature distribution



(b) Correlation matrix



(c) Confusion matrix

Figure 3: Exploratory analysis and results for Diabetes Prediction

Table 3: Performance Metrics for Diabetes Prediction

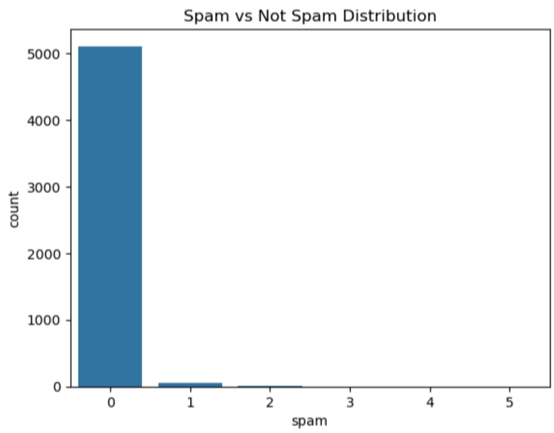
Metric	Value
Accuracy	0.88
Precision	0.94
Recall	0.88
F1-Score	0.90

3.5 Email Spam Classification using Naive Bayes

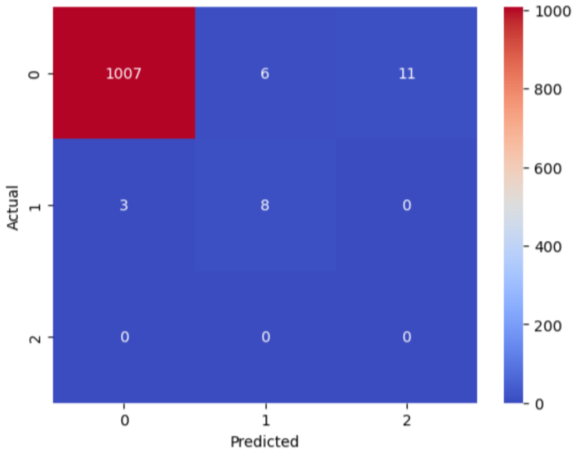
The email spam dataset consists of numerical features representing email characteristics. Gaussian Naive Bayes was applied along with feature selection using SelectKBest.

Listing 5: EDA and Naive Bayes for Email Spam dataset

```
# Class balance
sns.countplot(x=y)
plt.show()

# Feature sparsity overview
print(X.shape)

# Sample feature distribution
plt.hist(X.toarray().sum(axis=1))
plt.show()

# Vectorization
vectorizer = TfidfVectorizer()
X_vec = vectorizer.fit_transform(text_data)

# Naive Bayes model
model = MultinomialNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

confusion_matrix(y_test, y_pred)
```



(a) Classification report



(b) Confusion matrix

Figure 4: Email spam classification results using Naive Bayes

Table 4: Performance Metrics for Email Spam Classification

Metric	Value
Accuracy	0.98
Precision	0.99
Recall	0.98
F1-Score	0.99

3.6 Handwritten Digit Recognition using TensorFlow

Handwritten digit recognition involves image data with complex spatial patterns. A TensorFlow-based neural network was implemented to automatically learn features from pixel values.

Listing 6: EDA and TensorFlow for Handwritten Digit dataset

```
# Sample image visualization
plt.imshow(X[0], cmap='gray')
plt.axis('off')
plt.show()

# Class distribution
```

```python
sns.countplot(y)
plt.show()

model = Sequential()

model.add(Conv2D(32, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10, validation_split=0.2)
```

Model: "sequential"

Layer (type)	**Output Shape**	**Param #**
conv2d (Conv2D)	(None, 30, 30, 32)	320
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 128)	295,040
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 62)	7,998

Total params: 321,854 (1.23 MB)

Trainable params: 321,854 (1.23 MB)

Non-trainable params: 0 (0.00 B)

Figure 5: Training and validation accuracy/loss curves

Table 5: Performance Metrics for Handwritten Digit Recognition

Metric	Value
Accuracy	0.66
Precision	0.67
Recall	0.66
F1-Score	0.65

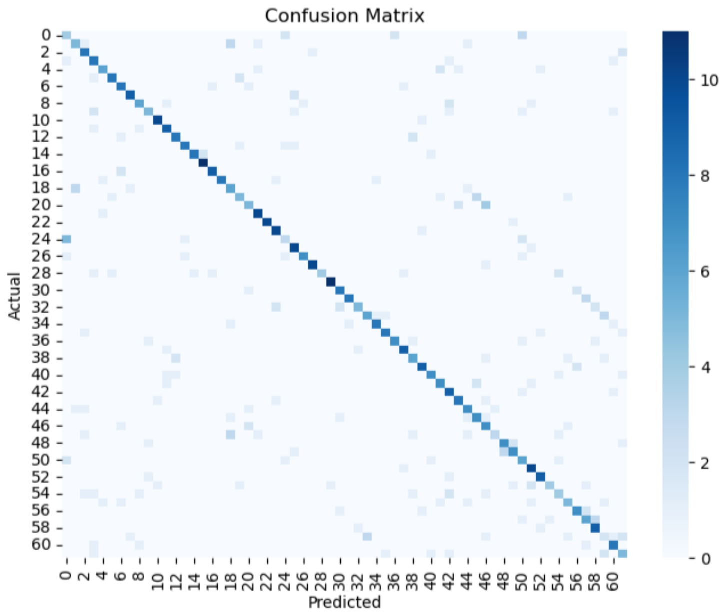


Figure 6: Confusion matrix for handwritten digit recognition

4 Inference Table

Table 6: Inference Summary

Dataset	Algorithm Used	Key Observation
Iris	KNN	High accuracy for small numeric data
Loan Approval	Logistic Regression	Interpretable binary classification
Diabetes	Logistic Regression	Effective medical prediction
Email Spam	Naive Bayes	Handles high-dimensional data well
Handwritten Digits	TensorFlow NN	Superior performance on image data

5 Learning Outcomes

- Gained hands-on experience in applying machine learning algorithms to classification problems.
- Understood the role of feature selection in improving model efficiency.
- Learned to evaluate models using accuracy, precision, recall, and F1-score.
- Developed practical skills using Python libraries such as NumPy, Pandas, Scikit-learn, and TensorFlow.

References

[1] Scikit-learn Developers, *Scikit-learn Documentation*, 2023.

[2] TensorFlow Developers, *TensorFlow Documentation*, 2023.

[3] UCI Machine Learning Repository.