Assignment-5

Zarin Akter (2011704042), Rashik Iram Chowdhury (2111336642) and Md. Mutasim Farhan (2013123642)
Emails: {zarin.akter@northsouth.edu}, {rashik.chowdhury@northsouth.edu}, {mutasim.farhan@northsouth.edu}

May 9, 2024

I. Introduction

In this project, we perform binary classification, multi-class classification, and regression tasks using a variety of machine learning algorithms. The classifiers and regressors investigated include ZeroR, OneR, K-Nearest Neighbors (KNN), Naive Bayes, Support Vector Machines (SVM) for classification, and Support Vector Regression (SVR) for regression.

- **ZeroR:** ZeroR is a baseline algorithm that predicts the majority class, ignoring all predictor features. It provides a baseline for accuracy comparison.
- **OneR:** OneR is also a baseline algorithm. It generates a single-level decision tree based on a single predictor attribute that best separates the classes. It provides a simple, interpretable model for comparison.
- **K-Nearest-Neighbors (KNN):** KNN is an instance-based learning algorithm that classifies new instances based on their similarity to the nearest neighbors in the training data. It can handle non-linear decision boundaries and is effective for many classification tasks.
- **Naive Bayes:** Naive Bayes is a probabilistic classifier that applies Bayes' theorem with the "naive" assumption of independence between features. It is computationally efficient and can perform well even with limited training data.
- **Support Vector Machines (SVM):** SVM is a powerful algorithm that finds the optimal hyperplane separating classes in a high-dimensional feature space. SVMs can handle non-linear decision boundaries and are effective for both binary and multi-class classification tasks.
- **Support Vector Regression (SVR):** An extension of SVMs for regression problems, SVR finds the optimal hyperplane that best fits the target variable within a certain threshold. It can handle non-linear relationships and is robust to outliers in the data.

We aim to evaluate and compare the performance of these algorithms on binary classification, multi-class classification, and regression problems using four diverse datasets. For the classification tasks, we evaluate the performance of each classifier using the accuracy metric. Accuracy measures the overall correctness of the classifier. For the regression task, we employ Support Vector Regression (SVR) and assess its performance using the mean squared error (MSE) metric. MSE quantifies the average squared difference between the predicted and actual values, providing a measure of the model's overall prediction accuracy. The datasets selected for this analysis cover a range of domains, including car evaluation, brain cancer detection, wage prediction, and credit risk assessment. By comparing the results across these diverse classifiers and regressors, we aim to gain insights into their relative strengths and weaknesses for different types of problems. This comprehensive evaluation will help inform the selection of appropriate algorithms for future classification and regression tasks.

The paper is structured into three main sections. Section [2] outlines the methodology for training all the models, including data preprocessing, Feature Scaling, and model evaluation. In Section [3], we present the experimental results and analyze the performance of the models across various datasets and tasks. Finally, Section [4] offers conclusions drawn from the findings and discusses the implications for practical applications.

II. Method

In this section, we sequentially implement the experiments with ZeroR, OneR, KNN, Naive Bayesian, SVM, and SVR classifiers on four datasets: Multi-class classification (Car Evaluation), Binary classification (Brain Cancer), and two Regression tasks (Credit and Wage). The datasets are split into train and test sets using a 80:20 ratio dataset. Evaluation metrics, including accuracy, confusion matrix, precision, recall, F-score, the precision-recall curve for binary classification, and average precision, recall, F-score, and mean-squared error (MSE) for multi-class classification and regression, are computed, which can be seen in section 4. Fig. 1 shows the workflow of this study.

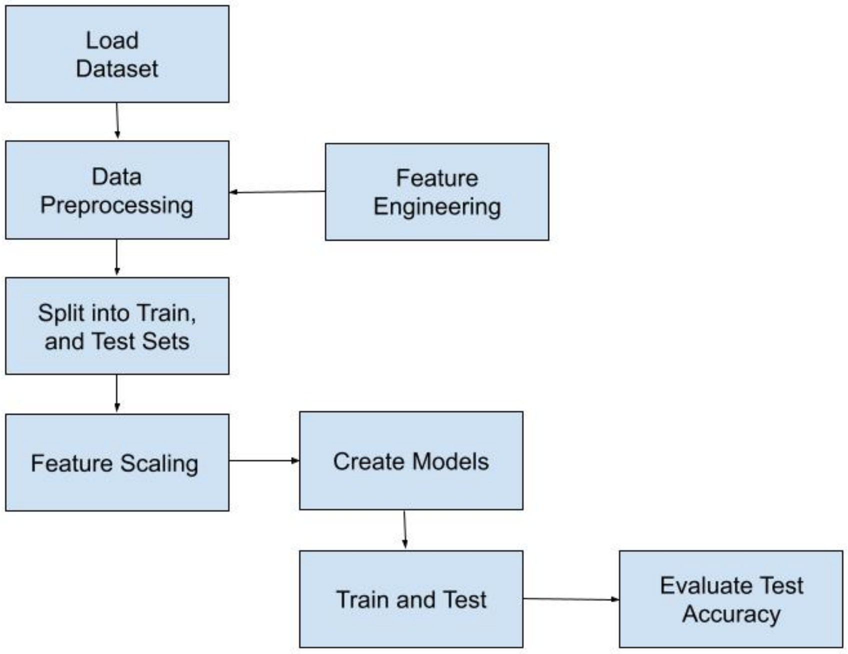

Fig. 1. Methodology Block Diagram

- **Load Dataset**

A. *Multi-Class Classification*

The dataset was obtained from the UCI Machine Learning Repository using the `fetch_ucirepo` function with the ID set to 19, corresponding to the Car Evaluation Dataset. The following code was used to fetch the dataset and load the features into `X` and the target into `y`:

```python
from ucimlrepo import fetch_ucirepo
import pandas as pd
import numpy as np
car_evaluation = fetch_ucirepo(id=19)

X = car_evaluation.data.features
y = car_evaluation.data.targets

print(car_evaluation.metadata)
print(car_evaluation.variables)
```

B. *Binary Classification*

The dataset used for Binary classification is DT-BrainCancer, which is in CSV format. The dataset is loaded using pandas.

```python
import pandas as pd
import numpy as np
df = pd.read_csv("G:\\Assignment\\DT-BrainCancer.csv")
df.head(5)
```

C. *Linear Regression*

i) The DT-Credit dataset is used for Linear Regression. It is in CSV format. The dataset is loaded using pandas.

```python
import pandas as pd
import numpy as np

df = pd.read_csv("G:\\Assignment\\DT-BrainCancer.csv")
```

ii) The DT-Wage dataset is also used for Linear Regression, which is loaded using pandas library.

```
import pandas as pd
import numpy as np

df = pd.read_csv("G:\\Assignment\\DT-Wage.csv")
```

- **Data Preprocessing:**

A. Multi-Class Classification

After loading the dataset and splitting it into X (features) and y (target), X and y are further preprocessed by converting the categorical values into numerical values using Label Encoding. This is because most models expect both features and target to be in numerical format.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X.loc[:, 'buying'] = le.fit_transform(X['buying'])
X.loc[:, 'maint'] = le.fit_transform(X['maint'])
X.loc[:, 'doors'] = le.fit_transform(X['doors'])
X.loc[:, 'persons'] = le.fit_transform(X['persons'])
X.loc[:, 'lug_boot'] = le.fit_transform(X['lug_boot'])
X.loc[:, 'safety'] = le.fit_transform(X['safety'])
y.loc[:, 'class'] = le.fit_transform(y['class'])
y = pd.DataFrame(y, columns=["class"])
y = y.astype(int)
```

B. Binary Classification

After loading the data frame, we found a missing value in the diagnosis column, for which that corresponding row or instance was dropped entirely. Then, 3 features - sex, diagnosis, and loc were converted to numerical values using Label Encoder. Then, the corresponding X and y are assigned to the features and target (status,) respectively.

```
df.dropna(subset=['diagnosis'], inplace=True)
df.isnull().sum()

df['sex'] = le.fit_transform(df['sex'])
df['diagnosis'] = le.fit_transform(df['diagnosis'])
df['loc'] = le.fit_transform(df['loc'])

X = df.drop(columns=['status'])
y = df['status']
```

C. Linear Regression

- For the Credit dataset, 4 features - Own, Student, Married, and Region are converted to numerical values using Label Encoder. Using a correlation heatmap, we found that the feature Own is weakly correlated. As a result, we dropped this column feature and processed X and y values by keeping and dropping the target (Balance), respectively.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Own'] = le.fit_transform(df['Own'])
df['Student'] = le.fit_transform(df['Student'])
df['Married'] = le.fit_transform(df['Married'])
df['Region'] = le.fit_transform(df['Region'])

X = df.drop(columns=['Balance', 'Own'])
y = df['Balance']
```

- For the Wage dataset, 7 features - maritl, race, education, region, jobclass, health, and health_ins are converted to numerical values using Label Encoder. Based on the correlation heatmap, we found that the region has no correlation with other features. As a result, it was dropped, and the remaining features were used to create X and y sets by selecting and dropping the target wage.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['maritl'] = le.fit_transform(df['maritl'])
df['race'] = le.fit_transform(df['race'])
df['education'] = le.fit_transform(df['education'])
df['region'] = le.fit_transform(df['region'])
df['jobclass'] = le.fit_transform(df['jobclass'])
df['health'] = le.fit_transform(df['health'])
df['health_ins'] = le.fit_transform(df['health_ins'])
X = df.drop('wage', axis=1)
y = df['wage']
```

- **Split into Train , and Test Sets:** We split the entire data into 80% train, and 20% test sets.This is applicable to all the datasets used.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

- **Feature Scaling** Standard Scalar is used to normalize the data. The code is applicable for both classification and regression tasks.

```
from sklearn.preprocessing import StandardScaler
scaler= StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

- **Create Model, Training and Testing:**

A. Classification Models:

After splitting into train and test sets, the following classifiers are invoked and trained with the training set for each dataset. After training the models, they are tested using the unseen test dataset for evaluation.

1) **ZeroR:**

```
from sklearn.dummy import DummyClassifier
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.dummy import DummyClassifier
model = DummyClassifier(strategy="most_frequent")
model.fit(X_train, y_train)
```

2) **OneR:**

```
from mlxtend.classifier import OneRClassifier
oner = OneRClassifier()

oner.fit(X_train_np, y_train)
```

3) **K-Nearest-Neighbour:**

```
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
```

4) **Naive Bayes:**

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)
```

5) **Support Vector Machine:**

```
from sklearn import svm
model = svm.SVC()
model.fit(X_train, y_train)
```

C. Regression Model

1) **Support Vector Regression:**

Support vector regression models were used for the Credit Dataset and Wage Dataset, where the best hyperparameters - Regularizer and epsilon- were changed manually to find the best MSE (mean squared error) for the unseen test dataset.

```
from sklearn import svm
from sklearn.metrics import mean_squared_error

model = svm.SVR(C=400, epsilon=0.2)
model.fit(X_train, y_train)
```

D. Evaluate Test Accuracy:

In this section, we present test accuracy evaluation for the six trained models using the accuracy metrics. The performance of each model is assessed on an unseen test dataset, and a comparative analysis is conducted to determine the effectiveness of the models.

The evaluation metrics employed for classification tasks include accuracy, confusion matrix, precision, and F-score. In regression tasks, we measure performance using mean squared error.

- **Accuracy**

In classification tasks, accuracy measures the percentage of correctly predicted instances out of all instances in the dataset. It is calculated as the number of correct predictions divided by the total number of predictions.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \times 100\% \tag{1}$$

- **Precision**

Precision is a metric that measures the accuracy of positive predictions made by a model. Precision is calculated as the ratio of true positive predictions to the total number of positive predictions made by the model.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2}$$

- **F1-Score**

The F1 score is a metric used to evaluate the model's accuracy, balancing precision, and recall. It considers false positives and negatives, making it a useful metric when the classes are imbalanced. The F1 score is calculated as the harmonic mean of precision and recall.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

- **Confusion Matrix**

A confusion matrix, also known as an error matrix, is a table that is used to evaluate the performance of a classification model. It presents a summary of the model's predictions against the actual outcomes across different classes in the dataset.

```
# Update confusion matrix
for true_label, pred_label in zip(y_test, y_test_predict):
    true_idx = np.where(unique_labels == true_label)[0][0]
    pred_idx = np.where(unique_labels == pred_label)[0][0]
    confusion_matrix[true_idx, pred_idx] += 1

print("Confusion␣Matrix:")
print(confusion_matrix)

import matplotlib.pyplot as plt
import seaborn as sns

```

```
# Plot Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix, annot=True, cmap=’Blues’, fmt=".0f")
plt.title(’Confusion␣Matrix’)
plt.xlabel(’Predicted␣Label’)
plt.ylabel(’True␣Label’)
plt.show()
```

III. Experiment results

A. Data Analysis

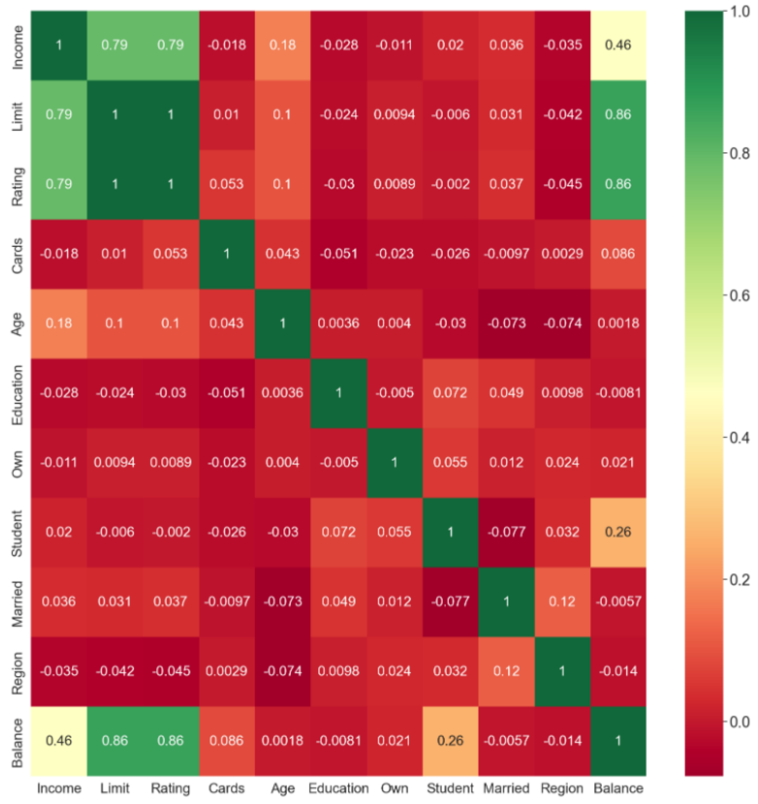

Fig. 2. Heat map of Regression Dataset DT-Credit

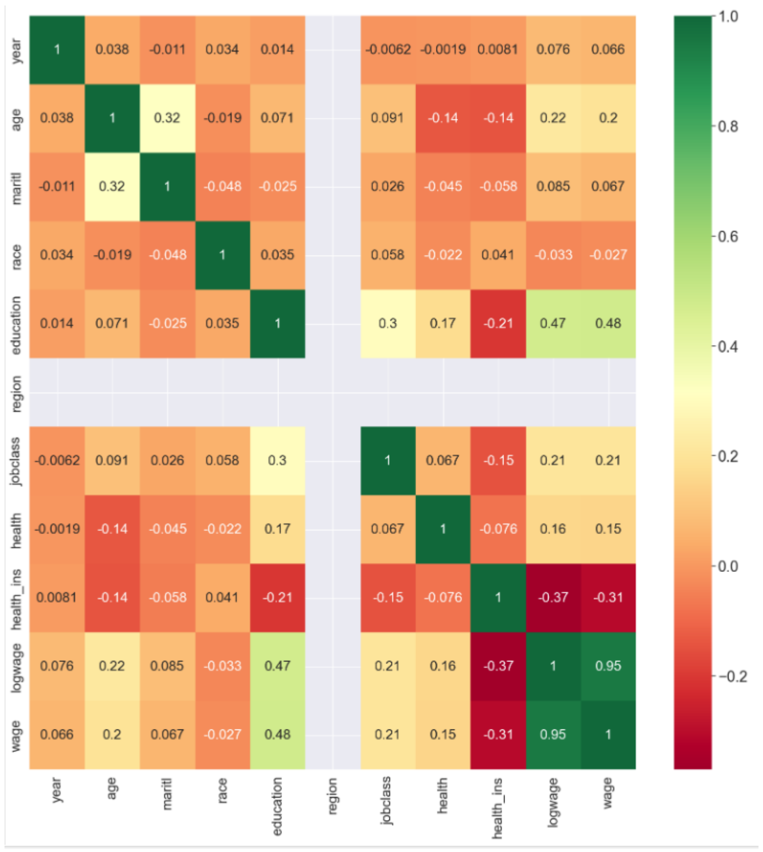

Fig. 3. Heat map of Regression Dataset DT-Wage

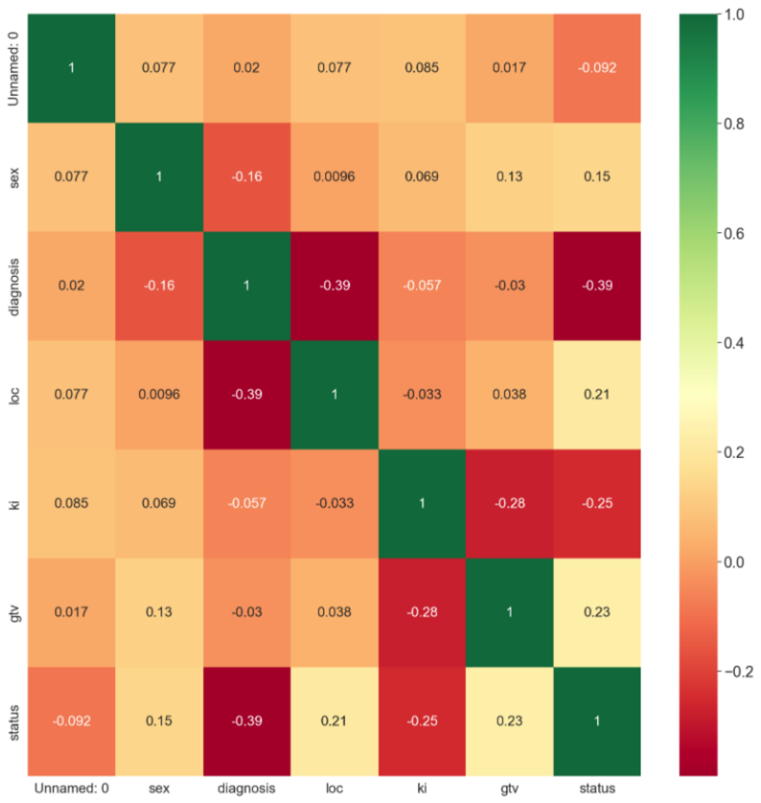

Fig. 4. Heat map of Binary Classification Dataset DT-BrainCancer

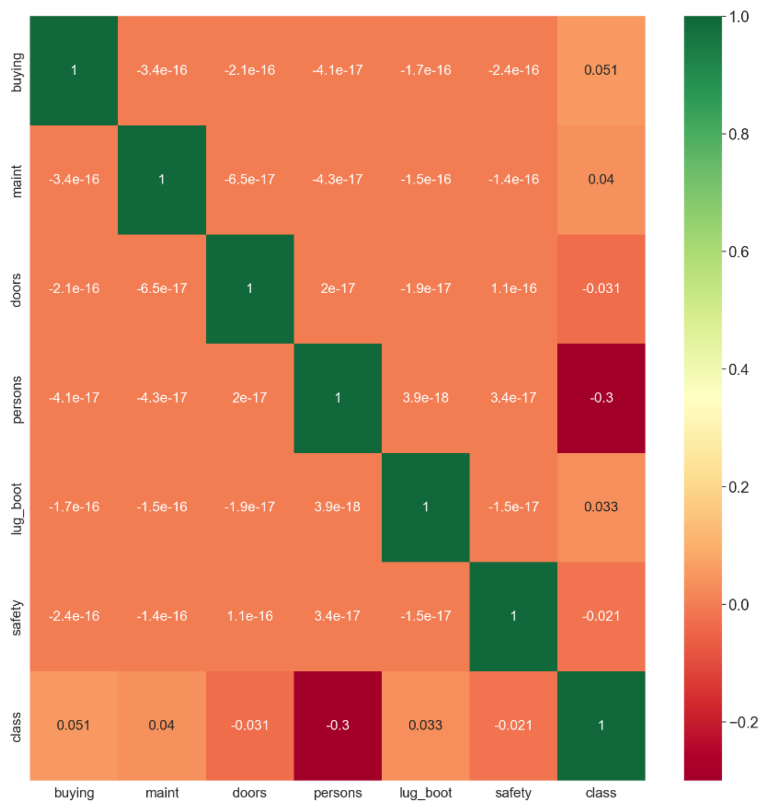

Fig. 5. Heat map of Multi-class classification Dataset

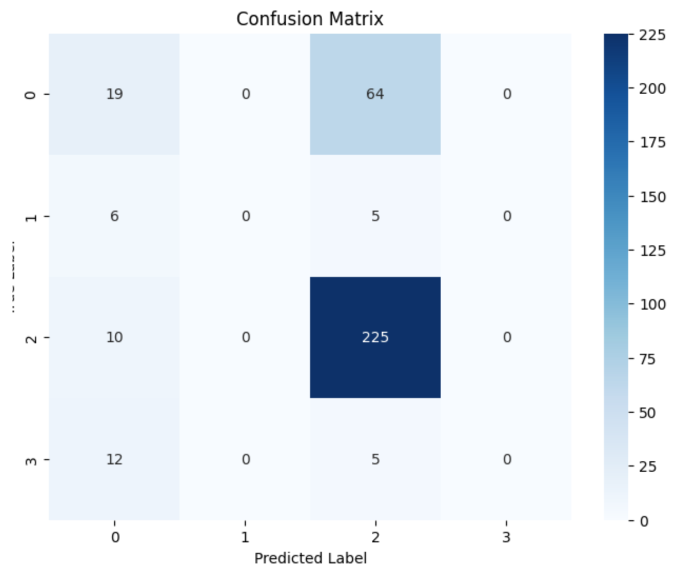

Fig. 6. Confusion matrix of Naive Bayes on Car Evaluation dataset (Multi-class classification)

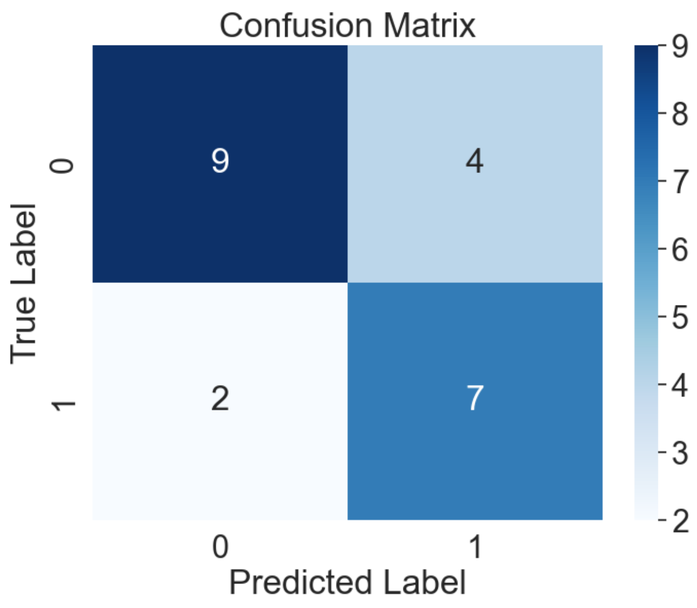

Fig. 7. Confusion matrix of Naive Bayes on Brain Cancer dataset (Binary classification)

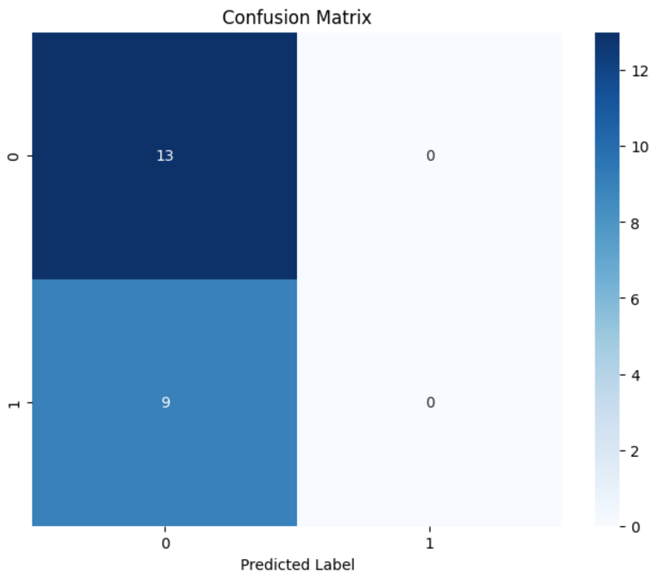

Fig. 8. Confusion matrix of OneR on Brain Cancer dataset (Binary classification)

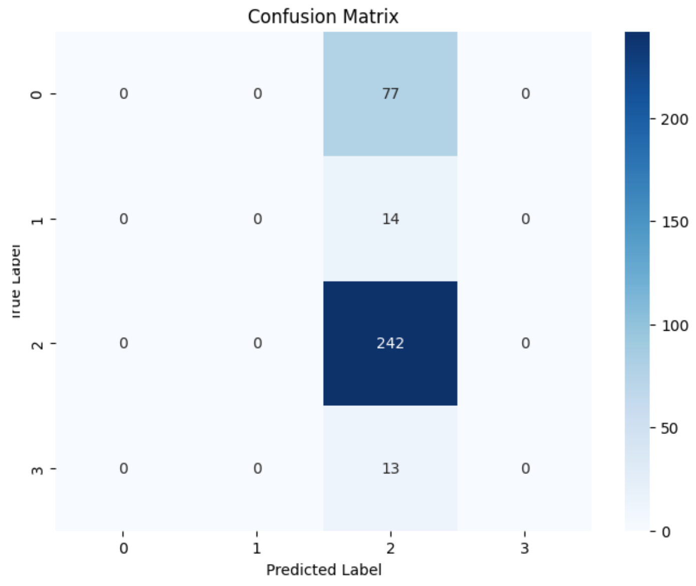

Fig. 9. Confusion matrix of OneR on Car Evaluation dataset (Multi-class classification)

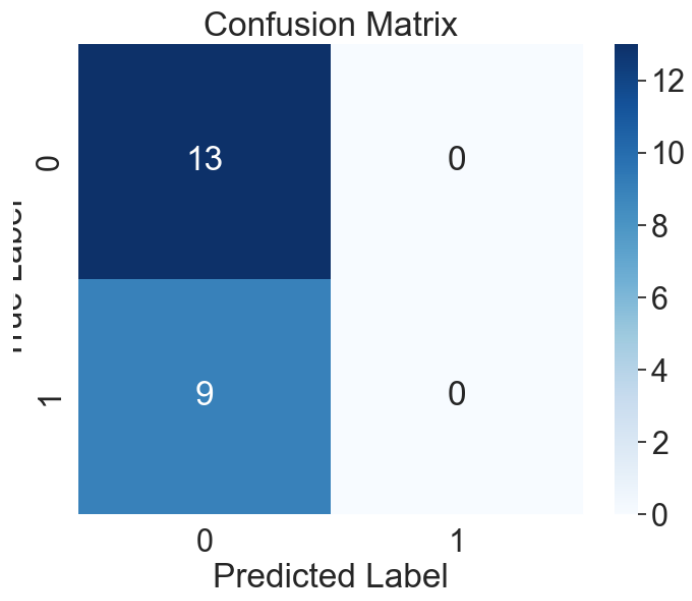

Fig. 10. Confusion matrix of ZeroR on Brain Cancer dataset (Binary classification)

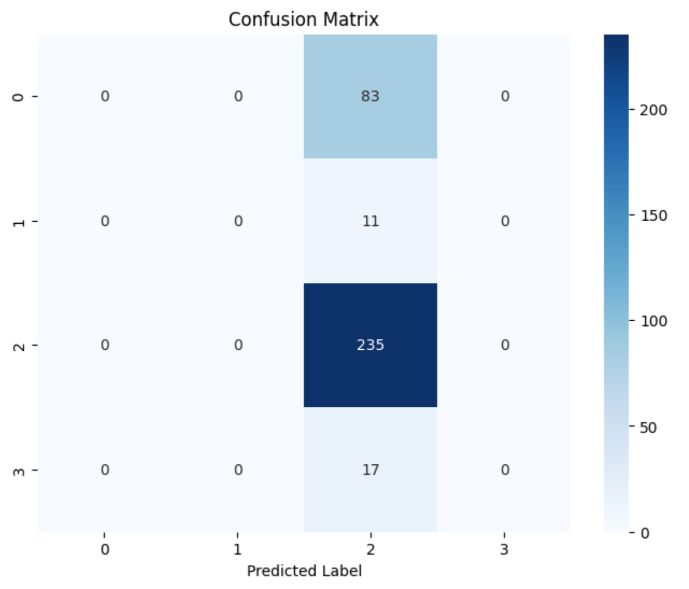

Fig. 11. Confusion matrix of ZeroR on Car Evaluation dataset (Multi-class classification)

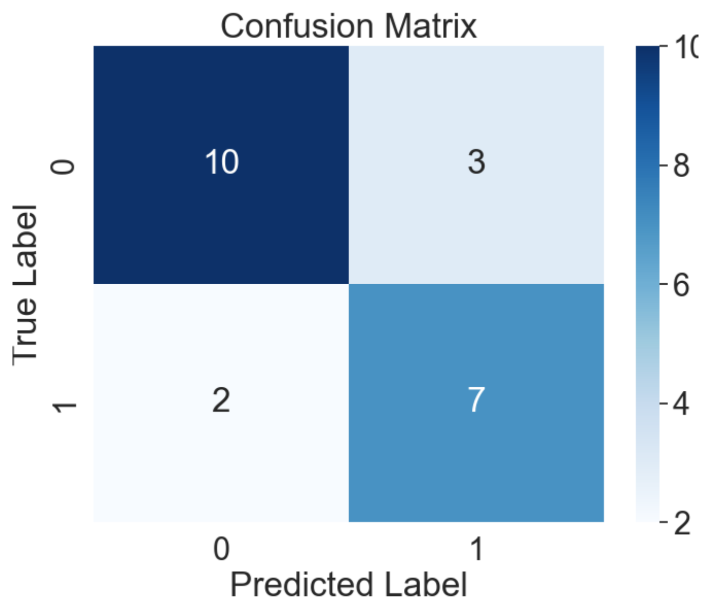

Fig. 12. Confusion matrix of Support Vector Machine on Brain Cancer dataset (Binary classification)

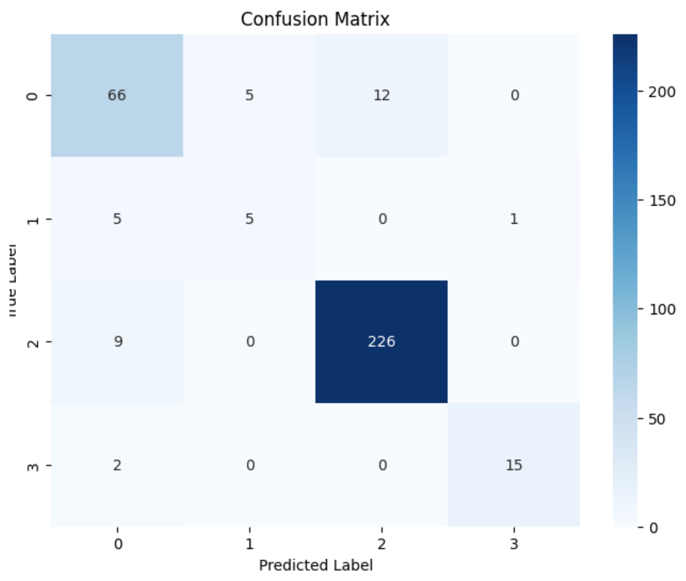

Fig. 13. Confusion matrix of Support Vector Machine on Car Evaluation dataset (Multi-class classification)

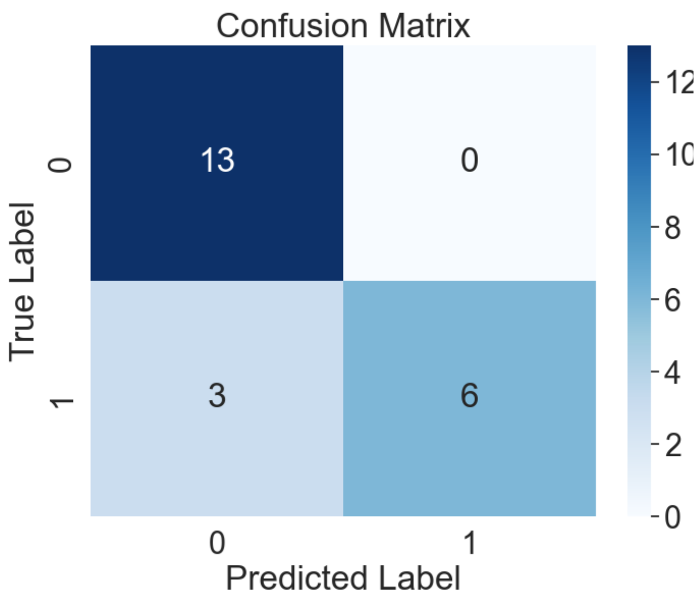

Fig. 14. Confusion matrix of K-Nearest-Neighbour on Brain Cancer dataset (Binary classification)

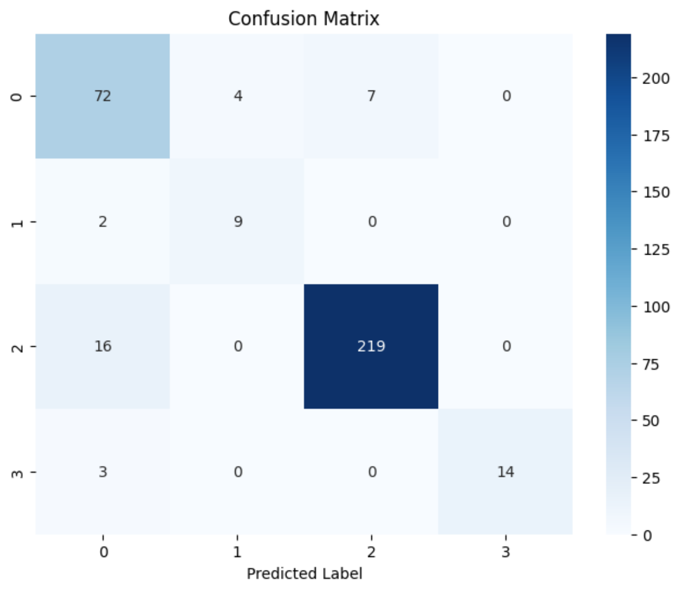

Fig. 15. Confusion matrix of K-Nearest-Neighbour on Car Evaluation dataset (Multi-class classification)

TABLE I
MODEL PERFORMANCE AND COMPARISON ANALYSIS ON BINARY AND MULTI CLASS CLASSIFICATION

Classification	**Dataset**	**Classifier Accuracy**				
		ZeroR	**OneR**	**KNN**	**Naive Bayesian**	**SVM**
Binary	Brain Cancer	59.09%	59.09%	86.36%	72.73%	77.27%
Multi-class	Car Evaluation	67.94%	69.94%	90.75%	70.52%	90.17%

TABLE II
MODEL PERFORMANCE AND COMPARISON ANALYSIS ON REGRESSION

Classification	**Dataset**	**Regression MSE**
		SVR
Regression	Credit	14878.19
	Wage	12.77

TABLE III
PERFORMANCE ANALYSIS OF NAIVE BAYESIAN

Naive Bayesian Classifier					
Naive Bayes Multi			**Naive Bayes Binary**		
Average Precision	**Average Recall**	**Average F1-Score**	**Average Precision**	**Average Recall**	**Average F1-Score**
0.2892	00.2966	0.2838	0.7273	0.7350	0.7250

TABLE IV
PERFORMANCE ANALYSIS OF ONER

OneR Classifier					
OneR Multi			**OneR Binary**		
Average Precision	**Average Recall**	**Average F1-Score**	**Average Precision**	**Average Recall**	**Average F1-Score**
0.1749	0.25	0.2058	0.2955	0.5	0.3714

TABLE V
PERFORMANCE ANALYSIS OF ZEROR

ZeroR Classifier					
ZeroR Multi			**ZeroR Binary**		
Average Precision	**Average Recall**	**Average F1-Score**	**Average Precision**	**Average Recall**	**Average F1-Score**
0.1698	0.25	0.2022	0.2955	0.5	0.3714

TABLE VI
PERFORMANCE ANALYSIS OF KNN

KNN Classifier					
KNN Multi			**KNN Binary**		
Average Precision	**Average Recall**	**Average F1-Score**	**Average Precision**	**Average Recall**	**Average F1-Score**
0.8589	0.8603	0.8554	0.9063	0.8333	0.8483

TABLE VII
PERFORMANCE ANALYSIS OF SVM

SVM Classifier					
SVM Multi			**SVM Binary**		
Average Precision	**Average Recall**	**Average F1-Score**	**Average Precision**	**Average Recall**	**Average F1-Score**
0.7980	0.7734	0.7852	0.7667	0.7735	0.7684

IV. DISCUSSION

This paper compares the baseline performance set by the baseline models - ZeroR and OneR classifiers - with the optimized models. Moreover, further comparison was made for each set of classification and regression datasets using the performance metrics defined above. distinguishing datasets comprising classification and regression problems. The models are first trained and then evaluated using the unseen test dataset.

Among both Binary and Multi-class Classification problems, we can see that KNN (K-Nearest-Neighbour) performed better, with around 86.36% and 90.75% accuracy, respectively, compared to five (5) other models. It is also important to note that all models demonstrated comparatively superior accuracy to the baseline accuracy of ZeroR and OneR.

In the case of Regression on both datasets, Support Vector Regression showed MSE (mean squared error), 14878.19 and 12.77 on the Credit and Wage datasets, respectively. However, Credit demonstrated high MSE, most likely because the model during training has been overfitted or there are not enough useful features.

REFERENCES

Bohanec, Marko. (1997). Car Evaluation. UCI Machine Learning Repository. https://doi.org/10.24432/C5JP48.