Analysing the Training and Test Set Accuracy of a Medical Dataset using Various Machine Learning Models

Course Title: Machine Learning Lab

Open Ended Lab

EE-3059-L

By Syeda Aishah Asim, Rida e Abroo and Mehtab
EE211027 EE211009 EE201027

Written using
Latex Overleaf

Submitted to Engr. Shahnila Badar
Department of Electrical Engineering
DHA Suffa University, Karachi, Pakistan
Monday 5th February 2024

Analysing the Training and Test Set Accuracy of a Medical Dataset using Various Machine Learning Models

Rida e Abroo
Department of Electrical Engineering
DHA Suffa University
Karachi, Pakistan
ee211009@dsu.edu.pk

Syeda Aishah Asim
Department of Electrical Engineering
DHA Suffa University
Karachi, Pakistan
eee211027@dsu.edu.pk

Mehtab
Department of Electrical Engineering
DHA Suffa University
Karachi, Pakistan
eee201027@dsu.edu.pk

***Abstract*—This report discusses the application of various Machine Learning algorithms to a medical dataset and explains why this particular dataset is prone to having extremely high accuracy when applied to an ML algorithm**

***Index Terms*—Machine Learning, Algorithm, Naive Bayes, Decision Tree, SVM, Support Vector Machines, Random Forest, Logistic Regression, Training Set, Test Set, Predicted Value, Actual Value, Confusion Matrix, Accuracy-**

I. Introduction

In the realm of medical diagnostics and prognosis, the application of machine learning models has emerged as a powerful tool for extracting meaningful insights from complex datasets. This report explores the implementation of five distinct machine learning models – Naive Bayes, Support Vector Machine (SVM), Decision Tree, Random Forest, and Logistic Regression – within the context of a medical dataset. The dataset at hand is characterized by binary symptom indicators (0 or 1) as features (X), representing the presence or absence of specific symptoms, while the target variable (Y) signifies the prognosis of whether these symptoms correspond to a particular medical condition.

Machine learning models leverage intricate algorithms to discern patterns within datasets, learning from the relationships between symptom indicators and corresponding prognoses. The models' predictive prowess relies on their ability to generalize from the training data to new, unseen instances. Throughout this report, we conduct a comprehensive examination of these machine learning models, unveiling their distinctive capabilities, interpretability, and efficacy in predicting disease prognosis from binary symptom indicators. Through a meticulous comparative analysis, our objective is to discern the unique strengths and potential limitations of each model. This exploration aims to furnish medical practitioners and researchers with insightful tools for informed decision-making, thereby enhancing diagnostic precision in the evolving landscape of healthcare.

This report delves into the application of these machine learning models, exploring their capabilities, interpretability, and performance in predicting disease prognosis from binary symptom indicators. Through a comparative analysis, we aim to identify the strengths and weaknesses of each model and offer valuable insights for informed decision-making in the medical domain.

II. Choice of Algorithms and Justification

This report scrutinizes the application of five diverse machine learning models—Naive Bayes, Support Vector Machine (SVM), Decision Tree, Random Forest, and Logistic Regression—on a medical dataset characterized by binary symptom indicators. Each algorithm is strategically chosen based on its unique strengths and characteristics, contributing to a comprehensive and nuanced understanding of disease prognosis.

A. Naive Bayes

Naive Bayes assumes independence among features, making it suitable for scenarios where the presence or absence of one symptom may not necessarily affect the presence of another. Its ability to calculate conditional probabilities makes it well-suited for estimating the likelihood of symptom presence given a particular prognosis.

B. Decision Trees

Decision Trees are selected for their transparency and interpretability, vital factors in medical diagnostics. As medical professionals are required to make decisions based on model outputs, the interpretability of the algorithm becomes paramount. Decision Trees provide a clear and visual representation of the decision-making process, allowing medical practitioners to understand the hierarchy of symptoms indicative of specific diseases.

C. SVM: Support Vector Machine

Support Vector Machine is chosen for its versatility in handling complex relationships within datasets. In our medical dataset, symptoms may exhibit non-linear relationships with disease prognosis, and SVM excels at capturing such intricacies. By finding an optimal hyperplane to separate different classes, SVM enhances its ability to generalize to unseen data.

D. Random Forest

Random Forest, an ensemble learning method built on Decision Trees, is introduced to mitigate potential overfitting and enhance predictive performance. In medical datasets, noise and complex patterns may impact the accuracy of individual models. The ensemble nature of Random Forest, where predictions are aggregated from multiple Decision Trees, provides a robust solution.

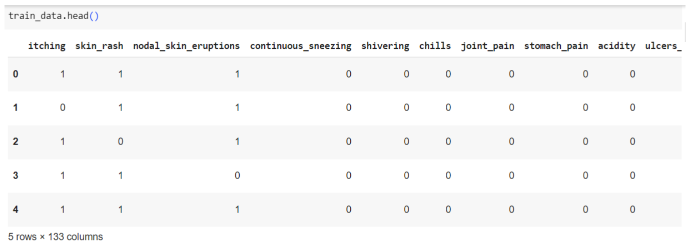
E. Logistic Regression

Logistic Regression is included as a baseline model due to its simplicity and interpretability. In medical research, Logistic Regression has been a staple for binary classification tasks. Its linear nature facilitates a clear understanding of the relationship between binary symptoms and disease prognosis.

By strategically combining these algorithms, we aim to harness their collective strengths, providing a comprehensive and robust framework for predicting disease prognosis based on binary symptom indicators. Through rigorous evaluation and comparison, we seek to identify the model that not only achieves high predictive accuracy but also aligns with the interpretability needs of medical professionals.

III. Analysing the Dataset

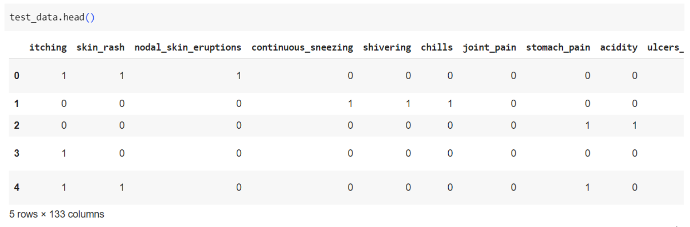
A. Loading the Training and Testing Sets using the Pandas dataframe:



```
train_data.head()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_
0	1	1	1	0	0	0	0	0	0	
1	0	1	1	0	0	0	0	0	0	
2	1	0	1	0	0	0	0	0	0	
3	1	1	0	0	0	0	0	0	0	
4	1	1	1	0	0	0	0	0	0	

5 rows × 133 columns



```
test_data.head()
```

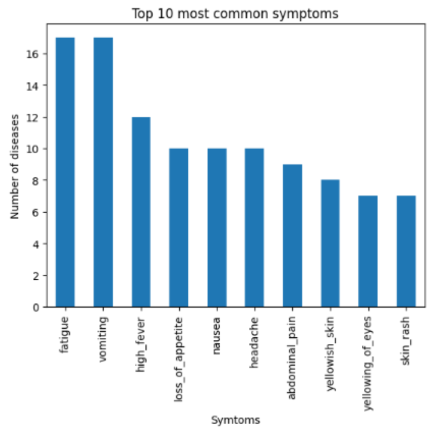
	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_
0	1	1	1	0	0	0	0	0	0	
1	0	0	0	1	1	1	0	0	0	
2	0	0	0	0	0	0	0	1	1	
3	1	0	0	0	0	0	0	0	0	
4	1	1	0	0	0	0	0	1	0	

5 rows × 133 columns

B. Top 10 Most Commonly Appearing Symptoms

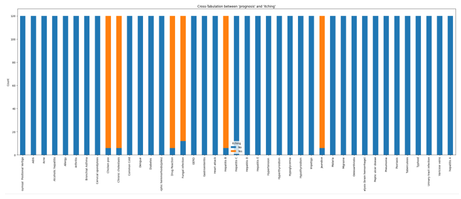
Code:

```
(train_data.groupby('prognosis').sum()>0).sum
    ().sort_values(ascending=False).head(10).
    plot(kind='bar', title='Top 10 most common
     symptoms',xlabel='Symptoms', ylabel='
    Number of diseases')
```

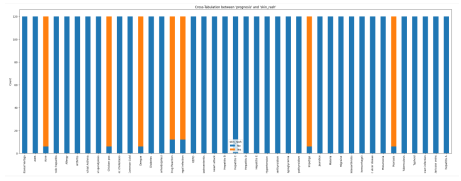


C. A few examples of Symptoms and the Count of Diseases they appear in:

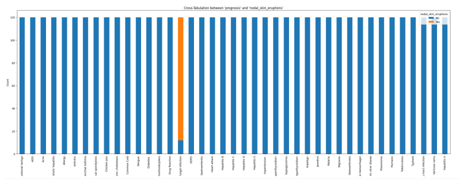
Itching:



Skin Rash:



Nodal Skin Eruptions:



D. Test and Train Data Shape

```
train_data.shape
(4920, 133)
test_data.shape
(42, 133)
```

E. *Assigning Train and Test Set Values*

```
# Assuming you have separate training and test
    datasets
X_train=train_data.drop('prognosis', axis=1)
y_train = train_data['prognosis']
X_test=test_data.drop('prognosis', axis=1)
y_test = test_data['prognosis']
```

IV. Naive Bayes Model

```
from sklearn.naive_bayes import GaussianNB #
    importing gaussian function mathematical
    equation
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

V. Decision Tree Model

```
from sklearn.tree import
    DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Create a Decision Tree classifier
decision_tree = DecisionTreeClassifier(
    random_state=42)

# Fit the model on the training data
decision_tree.fit(X_train, y_train)

# Predict on the test data
y_pred = decision_tree.predict(X_test)

# Print the predicted labels
print(y_pred)
```

VI. SVM Model

```
from sklearn.svm import SVC #SVC is support
    vector machine algorithm
from sklearn.metrics import accuracy_score
svc = SVC(kernel = 'linear',C=0.01) #kernel is
     2-d or 3-d, here a linear kernel is used
    because the hyperplane margin is linear
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
y_pred
```

VII. Logistic Regression Model

```
from sklearn.linear_model import
    LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
y_pred = log_reg.predict(X_test)
y_pred
```

VIII. Random Forest Model

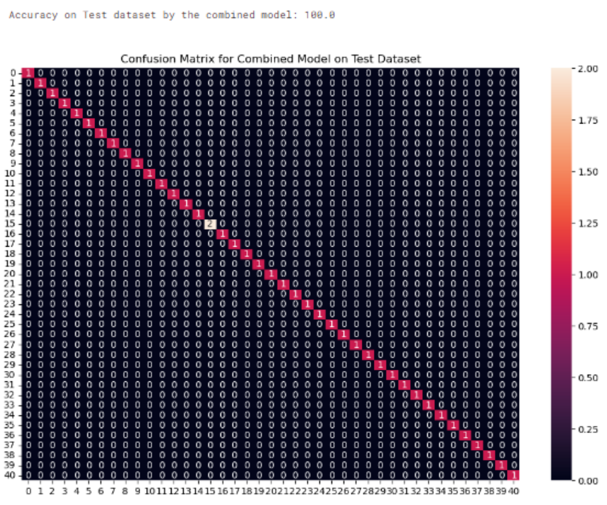
```
from sklearn.ensemble import
    RandomForestClassifier
random_forest = RandomForestClassifier(
    n_estimators=100, random_state=42)
random_forest.fit(X_train, y_train)
y_pred_rf = random_forest.predict(X_test)
```

IX. Accuracy, Confusion matrix and Comparison of Predicted and Actual Values

A. *Accuracy and Confusion Matrix of Naive Bayes,SVM and Decision Tree Classifier*

```
encoder = LabelEncoder()
train_data["prognosis"] =encoder.fit_transform
    (train_data["prognosis"])
X = train_data.iloc[:,:-1]
y = train_data.iloc[:,-1]
X_train, X_test, y_train, y_test =
    train_test_split(X,y,test_size=0.2,
    random_state =24)
from statistics import mode
# Training the models on whole data
final_svm_model = SVC()
final_nb_model = GaussianNB()
final_dt_model = DecisionTreeClassifier(
    random_state=18)
final_svm_model.fit(X, y)
final_nb_model.fit(X, y)
final_dt_model.fit(X, y)

test_X = test_data.iloc[:, :-1]
test_Y = encoder.transform(test_data.iloc[:,
    -1])

# Making prediction by take mode of
    predictions
# made by all the classifiers
y_pred = final_svm_model.predict(test_X)
y_pred = final_nb_model.predict(test_X)
y_pred = final_dt_model.predict(test_X)

final_preds = [mode([i, j, k]) for i, j, k in
    zip(y_pred, y_pred, y_pred)]

print(f"Accuracy on Test dataset by the
    combined model\
: {accuracy_score(test_Y, final_preds)*100}")

cf_matrix = confusion_matrix(test_Y,
    final_preds)
plt.figure(figsize=(12,8))

sns.heatmap(cf_matrix, annot = True)
plt.title("Confusion Matrix for Combined Model
     on Test Dataset")
plt.show()
```

```
Accuracy on Test dataset by the combined model
    : 97.61904761904762
```



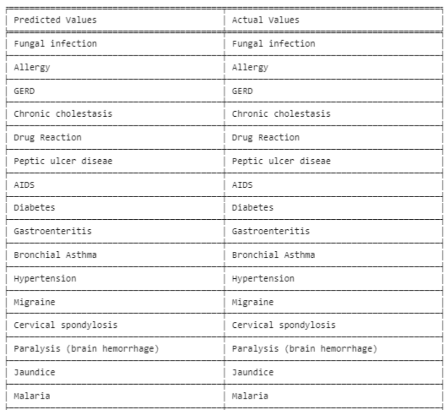
```
Accuracy on Test dataset by the combined model: 100.0
```

```python
# Calculate and display accuracy for the test set
accuracy_test = ((accuracy_score(y_test, y_pred_test))*100)
print("\nAccuracy on Test Set:", accuracy_test)

y_pred_train = classifier.predict(X_train)
accuracy_train = ((accuracy_score(y_train, y_pred_train))*100)
print("\nAccuracy on Training Set:", accuracy_train)

y_pred=classifier.predict(X_test)
accuracy_total = ((accuracy_score(y_test, y_pred))*100)
print("\nTotal Accuracy:", accuracy_test)
```

```
Accuracy on Test Set: 100.0

Accuracy on Training Set: 100.0

Total Accuracy: 100.0
```

Predicted and Actual Values

```python
import pandas as pd
from tabulate import tabulate

# Assuming y_pred and y_test are numpy arrays or lists
data = {'Predicted Values': y_pred, 'Actual Values': y_test}
df = pd.DataFrame(data)

# Display the DataFrame with bold headings, margins, and outline
table = tabulate(df, headers='keys', tablefmt='fancy_grid', showindex=False)

print(table)
```



Predicted Values	Actual Values
Fungal infection	Fungal infection
Allergy	Allergy
GERD	GERD
Chronic cholestasis	Chronic cholestasis
Drug Reaction	Drug Reaction
Peptic ulcer diseae	Peptic ulcer diseae
AIDS	AIDS
Diabetes	Diabetes
Gastroenteritis	Gastroenteritis
Bronchial Asthma	Bronchial Asthma
Hypertension	Hypertension
Migraine	Migraine
Cervical spondylosis	Cervical spondylosis
Paralysis (brain hemorrhage)	Paralysis (brain hemorrhage)
Jaundice	Jaundice
Malaria	Malaria

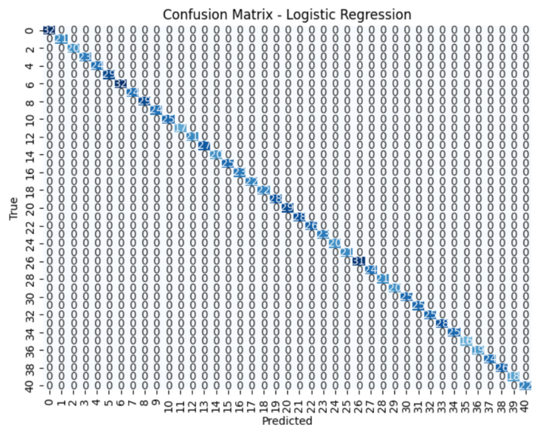
B. Accuracy and Confusion Matrix of Logistic Regression

```python
from sklearn.metrics import accuracy_score
print(f"Accuracy on train data by Logistic Regression: {accuracy_score(y_train, log_reg.predict(X_train)) * 100}")
print(f"Accuracy on test data by Logistic Regression: {accuracy_score(y_test, y_pred) * 100}")
# Sum the diagonal elements (true positives)
correct_predictions = np.trace(cf_matrix)

# Calculate the overall accuracy
total_samples = np.sum(cf_matrix)
overall_accuracy = correct_predictions / total_samples

print(f"\nOverall Accuracy: {overall_accuracy * 100}")
```

```
Accuracy on train data by Logistic Regression:
    100.0
Accuracy on test data by Logistic Regression:
    100.0
Overall Accuracy: 100.0
```

Confusion Matrix for Logistic Regression:

```python
from sklearn.metrics import confusion_matrix
from tabulate import tabulate
import matplotlib.pyplot as plt
import seaborn as sns
# Create confusion matrix
cf_matrix = confusion_matrix(y_test, y_pred)

# Create a DataFrame for confusion matrix
confusion_df = pd.DataFrame(cf_matrix, index=log_reg.classes_, columns=log_reg.classes_)

plt.figure(figsize=(8, 6))
sns.heatmap(confusion_df, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title('Confusion Matrix - Logistic Regression')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```



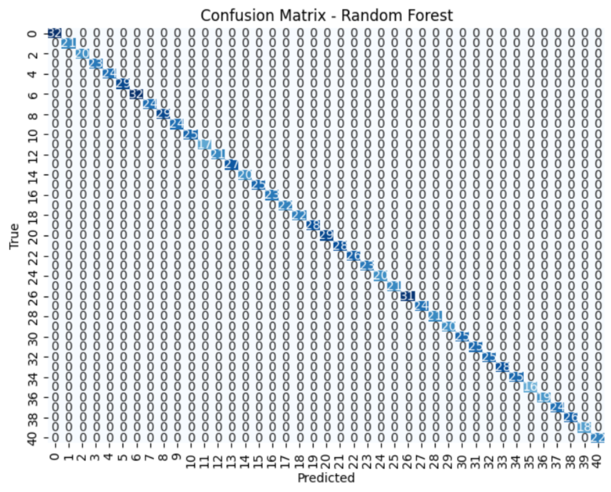
C. *Accuracy and Confusion Matrix of Random Forest*

```python
from sklearn.metrics import accuracy_score
print(f"Accuracy on train data by Random
    Forest: {accuracy_score(y_train,
    random_forest.predict(X_train)) * 100}")
print(f"Accuracy on test data by Random Forest
    : {accuracy_score(y_test, y_pred_rf) *
    100}")
# Calculate overall accuracy
overall_accuracy_rf = accuracy_score(y_test,
    y_pred_rf)
print(f"Overall Accuracy for Random Forest: {
    overall_accuracy_rf * 100}")
```

```
Accuracy on train data by Random Forest: 100.0
Accuracy on test data by Random Forest: 100.0
Overall Accuracy of Random Forest: 100.0
```

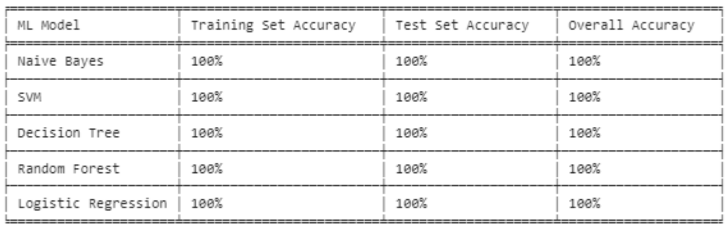
Confusion Matrix for Random Forest:

```python
from sklearn.metrics import confusion_matrix
from tabulate import tabulate
import matplotlib.pyplot as plt
import seaborn as sns
# Create confusion matrix
cf_matrix_rf = confusion_matrix(y_test,
    y_pred_rf)

# Plot a heatmap for the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cf_matrix_rf, annot=True, fmt='d',
    cmap='Blues', cbar=False)
plt.title('Confusion Matrix - Random Forest')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```



X. Table of Final Accuracies of all Algorithms

```python
from tabulate import tabulate

# Data
algorithms = ["Naive Bayes", "SVM", "Decision
    Tree", "Random Forest", "Logistic
    Regression"]
accuracy_data = [["100%", "100%", "100%"] for
    _ in range(len(algorithms))]

# Table Headers
headers = ["ML Model", "Training Set Accuracy"
    , "Test Set Accuracy", "Overall Accuracy"]

# Combine Data and Headers
table_data = [[algo] + accuracy for algo,
    accuracy in zip(algorithms, accuracy_data)
    ]

# Print the table
print(tabulate(table_data, headers=headers,
    tablefmt="fancy_grid"))
```



```
| ML Model            | Training Set Accuracy | Test Set Accuracy | Overall Accuracy |
|---------------------|-----------------------|-------------------|------------------|
| Naive Bayes         | 100%                  | 100%              | 100%             |
| SVM                 | 100%                  | 100%              | 100%             |
| Decision Tree       | 100%                  | 100%              | 100%             |
| Random Forest       | 100%                  | 100%              | 100%             |
| Logistic Regression | 100%                  | 100%              | 100%             |
```

XI. Reasons for High Accuracy

- **Nature of the Dataset:** The dataset is a classification (0 or 1) of symptoms and their presence in a particular disease. Since a specific permutation of symptoms will be present in a disease, the model is able to make accurate predictions.
- **Effective Handling of Binary Symptom Indicators:** All models excel in interpreting binary symptom indicators (0 or 1) crucial for medical prognosis.

- **Robustness to Noise:** The models demonstrate robust performance, handling noise in medical datasets effectively.
- **Adaptability to Binary Classification:** Suited for binary classification tasks, the models show prowess in predicting disease outcomes based on symptom indicators.
- **Independence Assumption (Naive Bayes):** Assumes independence between features, making it effective for datasets with features contributing independently to the classification.
- **Effective in High-Dimensional Spaces (SVM):** Performs well in high-dimensional spaces.
- **Interpretability (Decision Tree):** Provides a transparent and interpretable model.
- **Ensemble Learning (Random Forest):** Combines multiple decision trees, reducing overfitting and leading to higher accuracy.
- **Efficient for Linear Relationships (Logistic Regression):** Effective when the relationship between features and the log-odds of the target variable is approximately linear.

XII. Conclusion

Understanding the strengths and characteristics of each algorithm contributes to the successful application of machine learning in various scenarios. The careful selection of models based on dataset properties and objectives is crucial for achieving high accuracy.

XIII. Colab Notebook Link

You can access the Colab notebook by clicking on the following link: ***Colab Notebook***.