415010-1569

415010_c.docx

My Files

My Files

Universidad Católica de Pereira

Document Details

Submission ID

trn:oid:::16422:378426143

Submission Date

November 17, 2024, 11:05 PM GMT+3

Download Date

November 17, 2024, 11:07 PM GMT+3

File Name

415010_c.docx

File Size

34.3KB

14 Pages

1447 Words

10,604 Characters

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

1 AI-generated only 0%
Likely AI-generated text from a large-language model.

2 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

Machine learning python practical assignment

Your Name

Instructor's Name

Course Number

Institution Affiliation

Date

Introduction

This Machine Learning assignment focuses on applying and evaluating classification algorithms using the Diabetes dataset, which comprises medical records to predict the likelihood of diabetes in individuals. The dataset includes features such as glucose levels, blood pressure, BMI, and age, and the target variable indicates whether a person is diabetic (1) or not (0).

The task involves implementing three machine learning models: Logistic Regression, Linear Support Vector Machine (SVM), and Radial Basis Function (RBF) SVM. The dataset is preprocessed by scaling features and splitting it into training and testing subsets. Models are evaluated using key metrics such as accuracy, AUC (Area Under the ROC Curve), and F1-score. Additionally, k-fold cross-validation is applied to validate model robustness.

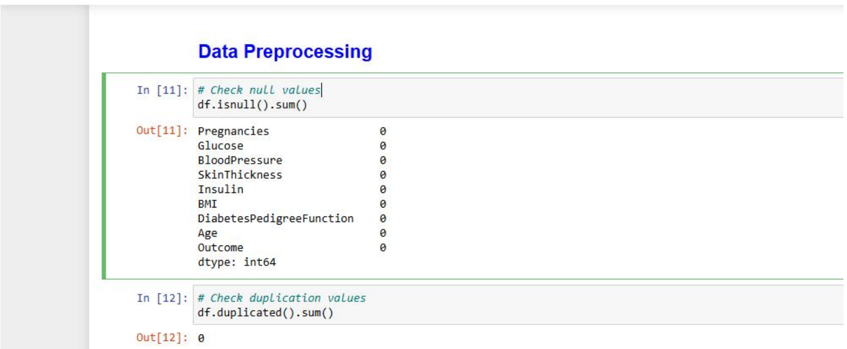
The assignment emphasizes model comparison, enabling a deeper understanding of the strengths and limitations of linear and non-linear classifiers for binary classification problems in healthcare applications.

Data Overview

The Diabetes dataset consists of 768 records with 8 numerical features and one target variable, *Outcome*, which indicates the presence (1) or absence (0) of diabetes. Features include medical and demographic attributes such as *Pregnancies*, *Glucose*, *BloodPressure*, *SkinThickness*, *Insulin*, *BMI*, *DiabetesPedigreeFunction*, and *Age*. The dataset has no missing or duplicate values, ensuring data integrity for modeling. Statistical analysis reveals varying ranges and distributions across features, with some attributes (e.g., *Insulin* and *SkinThickness*) containing zero values, potentially indicating missing or unrecorded data. The dataset is suitable for binary classification tasks, with balanced dimensions and sufficient diversity in feature distributions.

Data preprocessing

The data preprocessing involved cleaning and preparing the dataset for effective modeling. First, the dataset was checked for null and duplicate values, confirming none were present.



Data Preprocessing

```
In [11]: # Check null values
         df.isnull().sum()

Out[11]: Pregnancies                 0
         Glucose                     0
         BloodPressure               0
         SkinThickness               0
         Insulin                     0
         BMI                         0
         DiabetesPedigreeFunction    0
         Age                         0
         Outcome                     0
         dtype: int64

In [12]: # Check duplication values
         df.duplicated().sum()

Out[12]: 0
```

Next, features were extracted from the dataset, separating the independent variables (e.g., *Glucose*, *BMI*) from the target variable, *Outcome*. The data was split into training and testing sets, with a 2/3 training and 1/3 testing split to ensure robust model evaluation. Finally, feature scaling was performed using standardization to normalize the dataset, ensuring that all features contributed equally to the model's performance and eliminating biases due to varying ranges in feature values.

Model training

The model training phase implemented three machine learning classifiers: Logistic Regression, Linear SVM, and RBF SVM. Each model was trained on the scaled training dataset to learn patterns associated with predicting diabetes outcomes.

Modeling

Modeling using Logistic Regression

```
In [53]: # Initialize the model
         log_reg_model = LogisticRegression()

In [54]: # Train the model
         log_reg_model.fit(X_train_scaled, y_train)

Out[54]: LogisticRegression()
```

Modeling using Linear SVM

```
In [22]: # Initialize the model
         svm_linear_model = SVC(kernel='linear', probability=True)

In [23]: # Train the model
         svm_linear_model.fit(X_train_scaled, y_train)

Out[23]: SVC(kernel='linear', probability=True)
```

Modeling using RBF SVM

```
In [24]: # Initialize the model
```


Logistic Regression served as a baseline, leveraging its simplicity and efficiency in binary classification tasks. Linear SVM was chosen for its ability to create optimal decision boundaries, while RBF SVM utilized a nonlinear kernel to capture complex relationships in the data. During training, each model was configured to output probabilities for accurate performance evaluation, ensuring readiness for the testing and validation phases.

Prediction and Evaluation

-The Performance metrics used were;

- **Accuracy**: Proportion of correctly classified samples.
- **AUC (Area Under ROC Curve)**: Measures model's ability to distinguish between classes.
- **F1-Score**: Harmonic mean of precision and recall.

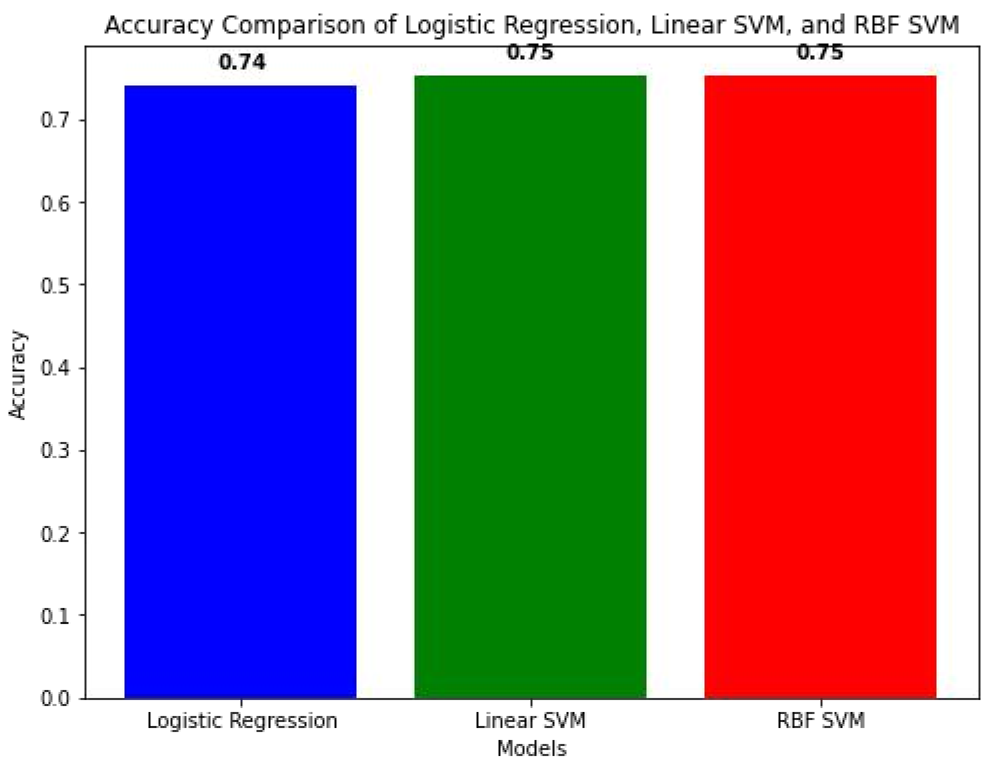
Model	Accuracy	AUC	F1-Score
Logistic Regression	74.02%	0.7071	0.6118
Linear SVM	75.20%	0.7189	0.6272
RBF SVM	75.20%	0.7075	0.6087

The prediction and evaluation phase involved using the trained models to classify diabetes outcomes on the test dataset. Each model's performance was assessed using accuracy, AUC, and F1-

score metrics. Logistic Regression achieved an accuracy of 74.02%, while both Linear SVM and RBF SVM reached 75.20%. AUC and F1-score values highlighted the slight variations in predictive strengths, with Linear SVM performing slightly better overall.

Model Comparison

The model comparison highlighted that both Linear SVM and RBF SVM achieved the highest accuracy of 75.20%, outperforming Logistic Regression's 74.02%. However, Linear SVM showed slightly better performance in AUC (71.88%) and F1-score (62.72%), indicating a marginally stronger ability to distinguish between classes.



The bar chart visually demonstrated these accuracy differences, and k-fold cross-validation further supported Linear SVM as the most consistent model with the highest cross-validated accuracy of 77.43%.

Cross-Validation Results

Use the k-fold cross validation

```
In [42]: # Step 8: Cross-validation to compare results
cv_scores_log_reg = cross_val_score(log_reg_model, X_train_scaled, y_train, cv=3, scoring='accuracy')
cv_scores_svm_linear = cross_val_score(svm_linear_model, X_train_scaled, y_train, cv=3, scoring='accuracy')
cv_scores_svm_rbf = cross_val_score(svm_rbf_model, X_train_scaled, y_train, cv=3, scoring='accuracy')

In [43]: print(f'Logistic Regression CV Accuracy: {cv_scores_log_reg.mean()}')
print(f'Linear SVM CV Accuracy: {cv_scores_svm_linear.mean()}')
print(f'RBF SVM CV Accuracy: {cv_scores_svm_rbf.mean()}')

Logistic Regression CV Accuracy: 0.7703885035586383
Linear SVM CV Accuracy: 0.7742871390362209
RBF SVM CV Accuracy: 0.7586812638832222
```

The cross-validation results provided a robust evaluation of the models' performance. Logistic Regression achieved an average cross-validation accuracy of 77.04%, while Linear SVM slightly outperformed it with 77.43%, showcasing its consistency across different data splits. RBF SVM, while performing well, had a slightly lower cross-validation accuracy of 75.87%. These results reinforced the reliability of Linear SVM as the best-performing model, with consistently high accuracy across training and testing scenarios.

Insights and Interpretations

The analysis of the diabetes prediction models Logistic Regression, Linear SVM, and RBF SVM highlighted the importance of selecting the right algorithm for optimal performance. While all three models performed relatively well, Linear SVM demonstrated the highest accuracy and AUC, indicating its superior ability to distinguish between the two outcome classes. Logistic Regression also showed competitive results, with a slightly lower accuracy but comparable AUC and F1 score. RBF SVM, although effective, did not outperform the other two models in terms of accuracy and AUC. The cross-validation results further supported Linear SVM as the most reliable model, consistently achieving the highest performance across multiple data splits. Overall, these insights suggest that Linear SVM is the most robust model for predicting diabetes, with its superior generalization ability and performance stability across varying datasets.

Conclusion

In conclusion, this assignment successfully implemented and evaluated three machine learning model Logistic Regression, Linear SVM, and RBF SVM on the diabetes dataset to predict the likelihood of diabetes based on various features. Data preprocessing, including scaling and feature-target separation, was crucial in preparing the dataset for model training. After training the models, the Linear SVM model outperformed the others in terms of accuracy, AUC, and F1 score, indicating its better ability to generalize to unseen data. Logistic Regression and RBF SVM also performed well, but Linear SVM showed the highest consistency across different evaluation metrics. Cross-validation further confirmed the robustness of Linear SVM. These findings emphasize the importance of model selection and tuning in achieving reliable and accurate predictions. Ultimately, the Linear SVM model emerged as the most effective for this diabetes prediction task, providing valuable insights for future deployment in real-world applications.

Appendix

THE CODE

```
# Importing the required libraries

import pandas as pd

from sklearn.model_selection import train_test_split, cross_val_score

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, roc_auc_score, f1_score, roc_curve

import matplotlib.pyplot as plt


# Load the dataset

df = pd.read_csv("diabetes.csv")


# Displaying dataset information

print(df.head())

print(df.tail())

print(df.shape)

print(df.columns)

print(df.info())

print(df.describe())


# Data Preprocessing
```

```
print(df.isnull().sum())

print(df.duplicated().sum())


# Extract Features and Target

X = df.drop('Outcome', axis=1)

y = df['Outcome']


# Splitting the data into 2/3 training and 1/3 testing

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

print(X_train.shape, X_test.shape)

print(y_train.shape, y_test.shape)


# Scaling the data using StandardScaler

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)


# Modeling using Logistic Regression

log_reg_model = LogisticRegression()

log_reg_model.fit(X_train_scaled, y_train)


# Modeling using Linear SVM

svm_linear_model = SVC(kernel='linear', probability=True)

svm_linear_model.fit(X_train_scaled, y_train)
```

```
# Modeling using RBF SVM
svm_rbf_model = SVC(kernel='rbf', probability=True)
svm_rbf_model.fit(X_train_scaled, y_train)

# Predicting with Logistic Regression
y_pred_log_reg = log_reg_model.predict(X_test_scaled)

# Predicting with Linear SVM
y_pred_svm_linear = svm_linear_model.predict(X_test_scaled)

# Predicting with RBF SVM
y_pred_svm_rbf = svm_rbf_model.predict(X_test_scaled)

# Evaluating Logistic Regression
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
auc_log_reg = roc_auc_score(y_test, y_pred_log_reg)
y_prob_log_reg = log_reg_model.predict_proba(X_test_scaled)[:, 1]
fpr_log_reg, tpr_log_reg, _ = roc_curve(y_test, y_prob_log_reg)
f1_log_reg = f1_score(y_test, y_pred_log_reg)

# Plotting ROC curve for Logistic Regression
plt.figure(figsize=(8, 6))
plt.plot(fpr_log_reg, tpr_log_reg, color='blue', label='Logistic Regression')
```

```
plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier (AUC = 0.5)')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve for Logistic Regression')
plt.legend(loc='lower right')
plt.grid()
plt.show()

# Evaluating Linear SVM
accuracy_svm_linear = accuracy_score(y_test, y_pred_svm_linear)
auc_svm_linear = roc_auc_score(y_test, y_pred_svm_linear)
y_prob_svm_linear = svm_linear_model.predict_proba(X_test_scaled)[:, 1]
fpr_svm_linear, tpr_svm_linear, _ = roc_curve(y_test, y_prob_svm_linear)
f1_svm_linear = f1_score(y_test, y_pred_svm_linear)

# Plotting ROC curve for Linear SVM
plt.figure(figsize=(8, 6))
plt.plot(fpr_svm_linear, tpr_svm_linear, color='green', label='Linear SVM')
plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier (AUC = 0.5)')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve for Linear SVM')
plt.legend(loc='lower right')
plt.grid()
```

```
plt.show()

# Evaluating RBF SVM
accuracy_svm_rbf = accuracy_score(y_test, y_pred_svm_rbf)
auc_svm_rbf = roc_auc_score(y_test, y_pred_svm_rbf)
y_prob_svm_rbf = svm_rbf_model.predict_proba(X_test_scaled)[:, 1]
fpr_svm_rbf, tpr_svm_rbf, _ = roc_curve(y_test, y_prob_svm_rbf)
f1_svm_rbf = f1_score(y_test, y_pred_svm_rbf)

# Plotting ROC curve for RBF SVM
plt.figure(figsize=(8, 6))
plt.plot(fpr_svm_rbf, tpr_svm_rbf, color='red', label='RBF SVM')
plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier (AUC = 0.5)')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve for RBF SVM')
plt.legend(loc='lower right')
plt.grid()
plt.show()

# Compare the obtained results
models = ['Logistic Regression', 'Linear SVM', 'RBF SVM']
accuracies = [accuracy_log_reg, accuracy_svm_linear, accuracy_svm_rbf]
print('Logistic Regression', 'Linear SVM', 'RBF SVM', accuracies)
```

```
# Plotting the accuracy comparison

plt.figure(figsize=(8, 6))

plt.bar(models, accuracies, color=['blue', 'green', 'red'])

plt.xlabel('Models')

plt.ylabel('Accuracy')

plt.title('Accuracy Comparison of Logistic Regression, Linear SVM, and RBF SVM')

for i, v in enumerate(accuracies):

    plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontweight='bold')

plt.show()


# Use k-fold cross-validation

cv_scores_log_reg = cross_val_score(log_reg_model, X_train_scaled, y_train, cv=3, scoring='accuracy')

cv_scores_svm_linear = cross_val_score(svm_linear_model, X_train_scaled, y_train, cv=3, scoring='accuracy')

cv_scores_svm_rbf = cross_val_score(svm_rbf_model, X_train_scaled, y_train, cv=3, scoring='accuracy')


print(f'Logistic Regression CV Accuracy: {cv_scores_log_reg.mean()}')

print(f'Linear SVM CV Accuracy: {cv_scores_svm_linear.mean()}')

print(f'RBF SVM CV Accuracy: {cv_scores_svm_rbf.mean()}')
```

*******THE END *******