A

MINI PROJECT REPORT

ON

"Liver Patients Detection Using Classification Model"

By

Mr. Darshan Kotkar	**(PRN No: 1954491245004)**
Mr. Vaibhav Pingle	**(PRN No: 1954491245025)**
Ms. Purva Patil	**(PRN No: 2054491245003)**
Ms. Mayuri Jadhav	**(PRN No:)**

T.Y. B. Tech (Computer Engineering), Semester -V

Department of Computer Engineering

Shri Vile Parle Kelavani Mandal's
Institute of Technology, Dhule – 424001

Affiliated to Dr. Babasaheb Ambedkar Technological University, Lonere.

Affiliated to Dr. Babasaheb Ambedkar Technological University, Lonere

[2021-22]

Department of Computer Engineering

SVKM's Institute of Technology, Dhule - 424001

CERTIFICATE

This is to certify that the Mini Project entitled **"Liver Patients Detection Using Classification Model"** has been carried out by Mr. Darshan Kotkar, Mr.Vaibhav Pingle, Ms. Purva Patil, Ms. Mayuri Jadhav, for Subject **Machine Learning** under the guidance of **Prof. Ashish Awate** in the premises of Department of Computer Engineering during the academic year 2021-22(Semester-V).

Date:

Place: Dhule

Mini Project Guide	Head of Department	Principal
Prof. Ashish Awate	Dr. Makarand Shahade	Dr. Nilesh Salunke

Shri Vile Parle Kelavani Mandal's

INSTITUTE OF TECHNOLOGY

Dhule (M.S.)

Subject : Machine learning.
Name : Darshan Nandkishor Kotkar Roll No.: 30
Class : T.Y. B.Tech Batch: T2 Division: C
Expt. No. : 09 Date : 08/01/2021
Title : Mini Project Report

Remark

Signature

Aim:- To Create model evaluation for reading and training dataset stored as .csv file and calculate accuracy for various models to conclude the best fitting model for given dataset

Software used : Google Colab, Python 3.

Dataset Used : Liver patients dataset containg Patient data in liver_patients .csv dataset

Observation & Conclusion : We have Successfully performed model evaluation on liver patients using classification models as Svm, KNN and Desecision tree.

The accuracy of classification model is nearly 66.67 %. the other evaluation matrix Score are

F1 score is 0.67 and

Jaccard Score is 0.68.

The result of our trained models are.

Algorithm	Accuracy	F1 Score	Jaccard Score
KNN	66.67%	0.67	0.68
SVM	65.59%	0.52	0.66
Decision Tree	66.67%	0.66	0.58

★ Team Members :

1] Darshan Kothkar [Roll No 30]
2] Vaibhav Pingale [Roll No 54]
3] Purva Padile [Roll No 45]
4] Mayuri Jadhav [Roll No 22]

Shri Vile Parle Kelavani Mandal's

INSTITUTE OF TECHNOLOGY

DHULE (M.S.)

DEPARMENT OF COMPUTER ENGINEERING

Subject: Machine Learning Lab

Name: Darshan Nandkishor Kotkar **Roll No.:** 30

Class: T.Y. Comp Engg. **Batch:** T2 **Division:** C

Expt. No.: 09 **Date:** 08/01/2022

Remark

Signature

Title: Mini Project: Create the Model Evaluation for a real life training data set stored as a .CSV file and by computing accuracy for different model proposed the best suitable machine learning algorithm for problem or solve the any real-life problem suggested by Hackerearth site

Aim:

The main aim of this lab is to demonstrate the how to Create the Model Evaluation for a real-life training data set stored

Secondary aim of this lab is to understand

- What are different evaluation metrics for classification and regression?
- How to evaluate machine learning model?
- How to propose the best fit model using different metrics?
- How to propose the best fit model using graphs?

Software Required:

Jupyter Notebook, Python 3.6, Or Google Colab

Dataset Used:

Liver_Patients.csv

Theory:

Metrics To Evaluate Machine Learning Algorithms in Python:

The metrics that you choose to evaluate your machine learning algorithms are very important.

Choice of metrics influences how the performance of machine learning algorithms is measured and compared. They influence how you weigh the importance of different characteristics in the results and your ultimate choice of which algorithm to choose.

In this practical, you will discover how to select and use different machine learning performance metrics in Python with scikit-learn.

There are two parts of evaluation depending upon your dataset and concept:
A) Evaluating Regression Models
B) Evaluating Classification Models

A) Evaluating Regression Models

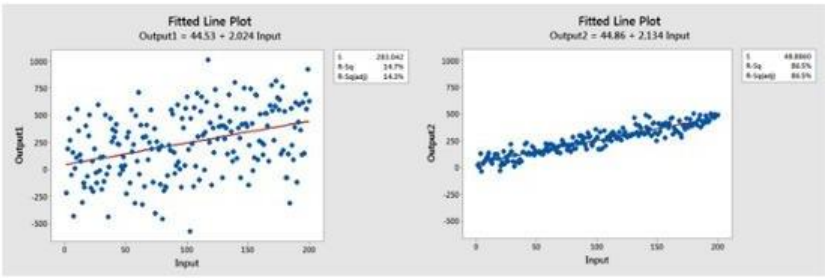
1. R-Squared

R Squared is a measurement that tells you to what extent the proportion of variance in the dependent variable is explained by the variance in the independent variables. In simpler terms, while the coefficients estimate trends, R-squared represents the scatter around the line of best fit.

For example, if the R^2 is 0.80, then 80% of the variation can be explained by the model's inputs.

If the R^2 is 1.0 or 100%, that means that all movements of the dependent variable can be entirely explained by the movements of the independent variables.

To show a visual example, despite having the same line of best fit, the R^2 on the right is much higher than the one on the left.



The equation for R^2 is as follows:

$$R^2 = 1 - \frac{Explained\ Variation}{Total\ Variation}$$

The Explained Variation is equal to the sum of squared residuals while the total variation is equal to the total sum of squared.

$$SS_{residual} = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2$$

$$SS_{total} = \sum_{i=0}^{n} (y_i - y_{avg})^2$$

Now that you understand what R^2 is, the code is very straight forward!

```
from sklearn.metrics import r2_score
sklearn.metrics.r2_score(y_true, y_pred)
```

2. Adjusted R-Squared

Every additional independent variable added to a model always increases the R^2 value — therefore, a model with several independent variables may seem to be a better fit even if it isn't. This is where Adjusted R^2 comes in. The adjusted R^2 compensates for each additional independent variable and only increases if each given variable improves the model above what is possible by probability.

There are a couple of ways to find the adjusted R^2 with Python:

Option 1: Manual Calculation

```
# n = number of sample size
# p = number of independent variablesAdj_r2 = 1-(1-R2)*(n-1)/(n-p-1)
```

Option 2: statsmodel.api

```
import statsmodels.api as sm
from statsmodels.sandbox.regression.predstd import wls_prediction_std
model1=sm.OLS(y_train,x_train)
result=model1.fit()
print(result.summary())
```

3. Mean Absolute Error (MAE)

The absolute error is the difference between the predicted values and the actual values. Thus, the mean absolute error is the average of the absolute error.

$$\text{Mean Absolute Error} = \frac{1}{N}\sum_{i=0}^{n} |y_i - \hat{y}_i|$$

By importing mean_absolute_error from sklearn.metrics, you can compute easily compute the MAE of your model.

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_true, y_pred)
```

4. Mean Squared Error (MSE)

The mean squared error or MSE is similar to the MAE, except you take the average of the squared differences between the predicted values and the actual values.

Because the differences are squared, larger errors are weighted more highly, and so this should be used over the MAE when you want to minimize large errors. Below is the equation for MSE, as well as the code.

$$\text{Mean Squared Error} = \frac{1}{N}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2$$

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_true, y_pred)
```

B) Evaluating Classification Models

1. Confusion Matrix and related metrics

A confusion matrix, also known as an error matrix, is a performance measurement for assessing classification models. Below is an example of a two-class confusion matrix.

	Predicted: No	Predicted: Yes
Actual: No	True Negative 50	False Positive 10
Actual: Yes	False Negative 5	True Positive 100

Within the confusion matrix, there are some terms that you need to know, which can then be used to calculate various metrics:

True Positive: Outcome where the model correctly predicts the positive class.
True Negative: Outcome where the model correctly predicts the negative class.
False Positive (Type 1 Error): Outcome where the model incorrectly predicts the positive class.
False Negative (Type 2 Error): Outcome where the model incorrectly predicts the negative class.

Now that you know these terms, here are a number of metrics that you can calculate:

Accuracy: equal to the fraction of predictions that a model got right.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

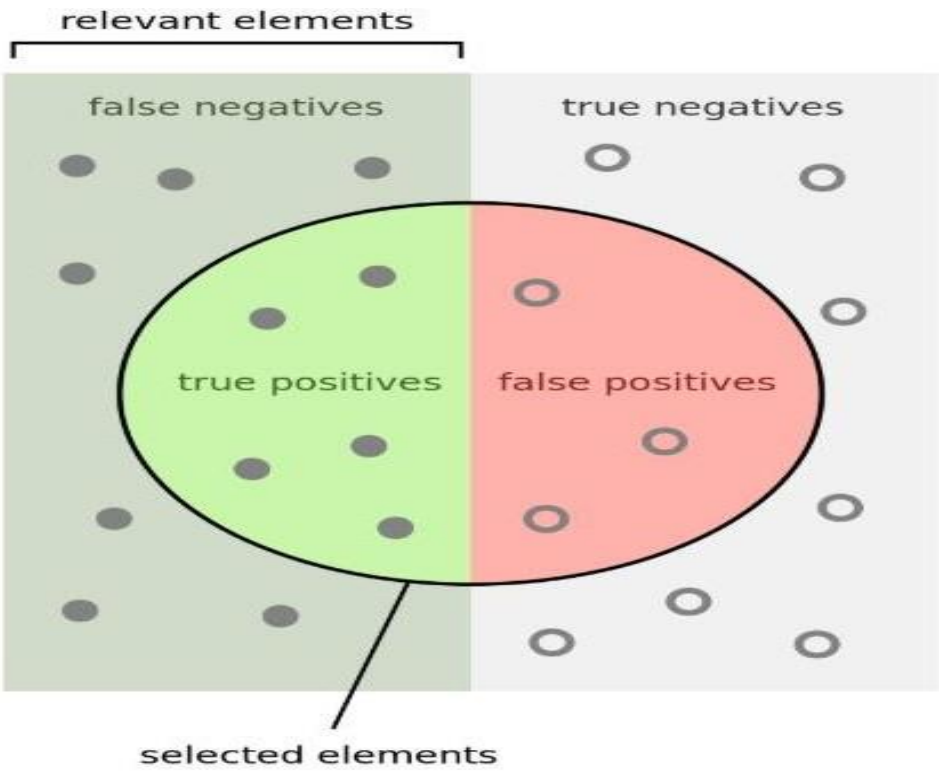
Recall: attempts to answer "What proportion of actual positives was identified correctly?"

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision: attempts to answer "What proportion of positive identifications was actually correct?"

$$\text{Precision} = \frac{TP}{TP + FP}$$

To really hit it home, the diagram below is a great way to remember the difference between precision and recall.



Code for confusion matrix and related metrics are below:

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_true, y_pred)

# Accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)

# Recall
from sklearn.metrics import recall_score
recall_score(y_true, y_pred, average=None)

# Precision
from sklearn.metrics import precision_score
precision_score(y_true, y_pred, average=None)
```

2. F1 Score:

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 * (precision * recall)}{precision + recall}$$

The F1 score is a measure of a test's accuracy — it is the harmonic mean of precision and recall. It can have a maximum score of 1 (perfect precision and recall) and a minimum of 0. Overall, it is a measure of the preciseness and robustness of your model.

There are three ways you can calculate the F1 score in Python:

```
# Method 1: sklearn
from sklearn.metrics import f1_score
f1_score(y_true, y_pred, average=None)

# Method 2: Manual Calculation
F1 = 2 * (precision * recall) / (precision + recall)

# Method 3: BONUS - classification report
from sklearn.metrics import classification_report
print(classification_report(y_true, y_pred,
target_names=target_names))
```

3. AUC-ROC Curve

The AUC-ROC Curve is a performance measurement for classification problems that tells us how much a model is capable of distinguishing between classes. A higher AUC means that a model is more accurate.

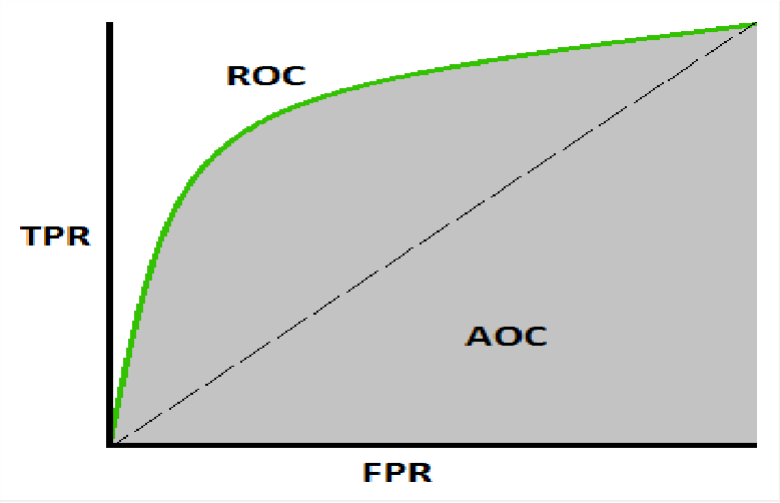
What is the AUC - ROC Curve?

AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.

The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.

If you want to learn more about it, Sarang does a great job explaining it here.

To calculate the AUC-ROC score, you can replicate the code below:



Defining terms used in AUC and ROC Curve.

TPR (True Positive Rate) / Recall /Sensitivity

$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN}$$

Specificity

$$\text{Specificity} = \frac{TN}{TN + FP}$$

FPR

$$\text{FPR} = 1 - \text{Specificity}$$

$$= \frac{FP}{TN + FP}$$

How to speculate about the performance of the model?

An excellent model has AUC near to the 1 which means it has a good measure of separability. A poor model has an AUC near 0 which means it has the worst measure of separability. In fact, it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s. And when AUC is 0.5, it means the model has no class separation capacity whatsoever.

To calculate the AUC-ROC score, you can replicate the code below:

```
import numpy as np
from sklearn.metrics import roc_auc_score
y_true = np.array([0, 0, 1, 1])
y_scores = np.array([0.1, 0.4, 0.35, 0.8])
roc_auc_score(y_true, y_scores)
0.75
```

4. Jaccard score

The Jaccard similarity index measures the similarity between two sets of data. It can range from 0 to 1. The higher the number, the more similar the two sets of data.

The Jaccard similarity index is calculated as:

Jaccard Similarity = (number of observations in both sets) / (number in either set)

Or, written in notation form:

$J(A, B) = |A \cap B| / |A \cup B|$

This tutorial explains how to calculate Jaccard Similarity for two sets of data in Python.
Jaccard similarity coefficient score

The Jaccard index [1], or Jaccard similarity coefficient, defined as the size of the intersection divided by the size of the union of two label sets, is used to compare set of predicted labels for a sample to the corresponding set of labels in ``y_true``.

Example: Jaccard Similarity in Python

```
from sklearn.metrics import jaccard_score
y_true = np.array([[0, 1, 1],
...                    [1, 1, 0]])
y_pred = np.array([[1, 1, 1],
...                    [1, 0, 0]])

In the binary case:
 jaccard_score(y_true[0], y_pred[0])   # doctest: +ELLIPSIS
0.6666...
In the multilabel case:
jaccard_score(y_true, y_pred, average='samples')
0.5833...
 jaccard_score(y_true, y_pred, average='macro')
0.6666...
jaccard_score(y_true, y_pred, average=None)
array([0.5, 0.5, 1. ])
```

5. Log_loss

Log loss, aka logistic loss or cross-entropy loss.

This is the loss function used in (multinomial) logistic regression and extensions of it such as neural networks, defined as the negative log-likelihood of a logistic model that returns **y_pred** probabilities for its training data **y_true**. The log loss is only defined for two or more labels. For a single sample with true label y∈{0,1} and a probability estimate p=Pr(y=1), the log loss is:

$$L_{\log}(y, p) = -(y\log(p) + (1 - y)\log(1 - p))$$

Examples

```
from sklearn.metrics import log_loss

log_loss(["spam", "ham", "ham", "spam"],

[[.1, .9], [.9, .1], [.8, .2], [.35, .65]])

0.21616...
```

Procedure	Import Libraries Import the Dataset. 1. Do the Exploratory data analysis 2. Do the preprocessing of data 3. Apply the classification/regression) models (min 3 and max 5) do you want wish to train your data 4. Do the model evaluation applicable to your model 5. Create the evaluation table 6. Visualize the evaluation model 7. Do the cross validation and proposed the best model for your dataset 8. If you are using classification build the ROC curve and discuss it 9. If you are using Regression model build the residual chart. 10. Conclude with best model for your dataset.

Observation & Conclusion	In this way, we successfully perform the model evaluation on:
	
	`Liver_Patients.csv`
	
	The best model for our dataset is:
	
	`K-Nearest NEIGHBOR.`
	
	With the following metrics of evaluation:
	
	`Accuracy:66.67%`
	Jaccard Score:0.68
	F1_Score:0.67
	
	The error values are less compared to other classification models
	
	ROC Curve/ Residual chart suggest us:
	
	The Residual chart suggests that classification training and testing data doesn't have much error difference from the hypothesis line. In fact, the training data lies exactly on the hypothesis line and the testing data slightly has a positive error difference
	Decision tree classification has a greater error difference in its testing data SVM also deviates with major error differences from the hypothesis line in its's testing set. Therefore, we conclude, with minimum error difference from hypothesis line,KNN is our best model for the salary prediction in our dataset.

Evolution Table

PERFORMANCE (OUT OF 3)	Write up (OUT OF 3)	Attendance (OUT OF 2)	Oral (OUT OF 2)	Total (OUT OF 10)