PULSAR STAR PREDICTION

Manogna Teja Pidikiti, Vyshnavi Aitu
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
mpidiki1@umbc.edu, vyshnaa1@umbc.edu

Pulsars are a rare type of Neutron star that produce radio emission detectable here on Earth. They are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter. Generally, pulsar search involves looking for periodic radio signals with large radio telescopes. Each pulsar produces a slightly different emission pattern, which varies slightly with each rotation. Thus, a potential signal detection known as a 'candidate', is averaged over many rotations of the pulsar, as determined by the length of an observation. In the absence of additional info, each candidate could potentially describe a real pulsar. So, in this project we have decided to apply several classifiers and predict the model which has a high accuracy. There are several projects which were done using the classifiers like kNN, linear regression for which the accuracy obtained was moderate. Hence, we will not use those classifiers but improve the accuracy by using support vector machines, decision trees and random forest classifiers.

I. INTRODUCTION

A pulsar star is a highly magnetized rotating neutron star that emits beams of electromagnetic radiation out of its magnetic poles. This radiation can be observed only when a beam of emission is pointing toward Earth (much like the way a lighthouse can be seen only when the light is pointed in the direction of an observer), and is responsible for the pulsed appearance of emission. Neutron stars are very dense, and have short, regular rotational periods. These pulsar stars have a great scientific advantages of looking for periodic radio signals with large radio telescopes. Every star has its own pace of rotation varying from pattern to pattern. Hence, a potential signal is found by the average of several rotations of the pulsar. In this project we would like to predict a pulsar star based on several features like Mean of the integrated profile, Standard deviation of the integrated profile, Excess kurtosis of the integrated profile, Skewness of the integrated profile, Mean of the DM-SNR curve etc. The dataset we are using is of HTRU2 which depicts the pulsar candidates captured during a survey. There are previous models which are built using linear regression classifier, kNN. In our project we would like to enhance the accuracy of the model using other classifiers like support vector machines, decision trees, random forests for the efficient accuracy and compare them with the previous models. The input to our model is HTRU2 dataset that contains an eight dimension vector which is pre-processed by UCI from some raw signal data. The model which we built will finally result in either 0 or 1 which indicates no pulsar star or the presence of pulsar star respectively. We measure the performance of the models based on the accuracy, precision, recall and AUC-ROC curve for both positive and negative labels.

II. PRE-PROCESSING OF DATA

The data which we have taken was already pre-processed but several aspects were taken care of. Initially, we have checked if the data had null values and also the range of the values is quite optimum with the same datatype or not throughout. Also, we have analysed the values of every column to find if the values are regular and the difference between the values is not high in order to avoid the outliers. The dataset which we have chosen surprisingly has no missing values in it and without any glitches. Initially, we noticed that the names of the attributes are very long, so we have renamed the attributes with much simpler names. The columns in the dataset are all essential to get the output, so none of the columns are dropped. We dropped the target class column from the dataset and assigned it to an other variable for our convenience. We utilised a special function to know the importance of each feature using the **feature importance** function.

III. RELATED WORK

The main challenges for the search of pulsars in the past fifty years is the analysis required for the increase of the 'candidate' pulsar detection due to the large volume of data that has been searched. Basically, our project mainly focuses on the diverse machine learning techniques to maximize the accuracy, precision and recall for both the positive and negative labels. In this section we would discuss several prior works that has been done on this dataset.

There is a **baseline** with the implementation of the same classifiers but achieved a low accuracy than our model in both the training and testing phases. The difference in the accuracy of the model is due to the changes in the split ratio, the value of the number of iterations used and the value of the n_estimators. The Reference of our baseline is provided in the section of references.

There have been several models which have been used to predict the pulsar stars on this dataset such as kNN, decision trees which achieved the accuracy to some extent but not to the extent as much as the models of support vector machines, logistic regression and random forests. So, in our project we

would compare the values of each classifier on the basis of accuracy, precision, recall and decide the appropriate classifier that can be used to predict the data. The above mentioned parameters can be used as the performance measure for the project along with the AUC-ROC curve.

IV. DATASET AND FEATURES

The Dataset used is of HTRU2 which is in the format of .csv(comma separated values). There are a total of 17,898 examples out of which 1,639 are positive examples comprising of 10 percent and 16,259 negative examples comprising of 90 percent. Each sample of candidate pulsar has eight continuous variables and a target class variable. The eight features are Mean of the integrated profile, Standard deviation of the integrated profile, Excess kurtosis of the integrated profile, Skewness of the integrated profile. Mean of the DM-SNR curve, Standard deviation of the DM-SNR curve, Excess kurtosis of the DM-SNR curve, Skewness of the DM-SNR curve and target class. Candidates are generally based on whether a pulsar star exists or not.

The first four statistics are obtained from the integrated pulsar profile. The integration or averaging of the received signal over a period of time will reduce the random noise related to the signal. The signal-to-noise ratio increases with the integration time. The above four statistics are an array of continuous variables which are the longitude-resolved version of the signal that have been averaged in both time and frequency. The other four are determined from the dispersion measure(DM) (A measurement of the delay experienced by the pulsar signal as it transits the interstellar medium. It is affected by the electron density along the line of sight. The units of dispersion measure are parsecs per cubic centimeter) and signal-to-noise ratio curve(DM-SNR) curve.The SNR measures the level of signal and the level of noise. DM-SNR curve is a combination between the real and theoretical acceleration-SNR curve. All of the features are radio emission-related physics properties and thus are appropriate for predicting the existence of pulsar stars.

Table 2: The eight features derived from the integrated pulse profile P and DM-SNR curve D from Lyon 2016[4]

Feature	Description	Definition
$Prof.\mu$	Mean of P.	$\frac{1}{n}\sum_{i=1}^{n} p_i$
$Prof.\sigma$	Standard deviation of P.	$\sqrt{\frac{\sum_{i=1}^{n}(p_i-\bar{P})^2}{n-1}}$
$Prof.k$	Excess kurtosis of P.	$\frac{\frac{1}{n}(\sum_{i=1}^{n}(p_i-\bar{P})^4)}{(\frac{1}{n}(\sum_{i=1}^{n}(p_i-\bar{P})^2))^2} - 3$
$Prof.s$	Skewness of P.	$\frac{\frac{1}{n}(\sum_{i=1}^{n} p_i-\bar{P})^3}{(\sqrt{\frac{1}{n}(\sum_{i=1}^{n}(p_i-\bar{P})^2)})^3}$
$DM.\mu$	Mean of D.	$\frac{1}{n}\sum_{i=1}^{n} d_i$
$DM.\sigma$	Standard deviation of D.	$\sqrt{\frac{\sum_{i=1}^{n}(d_i-\bar{D})^2}{n-1}}$
$DM.k$	Excess kurtosis of D.	$\frac{\frac{1}{n}(\sum_{i=1}^{n}(d_i-\bar{D})^4)}{(\frac{1}{n}(\sum_{i=1}^{n}(d_i-\bar{D})^2))^2} - 3$
$DM.s$	Skewness of D.	$\frac{\frac{1}{n}(\sum_{i=1}^{n} d_i-\bar{D})^3}{(\sqrt{\frac{1}{n}(\sum_{i=1}^{n}(d_i-\bar{D})^2)})^3}$

Fig. 1. **Values of each feature in the dataset**

```
train_x, test_x, train_y, test_y = train_test_split(X, Y, test_size = 0.3, random_state = 42)
```

Fig. 2. **Train and Test split ratio**

V. SPLITTING THE DATA

We have split our data into 70 percent training data and 30 percent testing data. This ratio was taken as it displayed the maximum performance in this measure.

VI. METHODS

A. *Logistic Regression*

In this project, we have utilised the model of linear equations with the independent predictors to classify the value of the target class variable. Generally, the predicted value can be between negative infinity or the positive infinity. So, the value of the output would be either 0 or 1. Hence, the output comes in the range of [0,1]. We have used a sigmoid function to predict the value as 0 or 1.

*1) **Evaluation of logistic regression results**:* Generally, the evaluation of the machine learning algorithm is the essential part of finding the performance measure. Here we can use the accuracy of the model as the performance measure. This can also be called as the classification accuracy or the accuracy score. The terms means that it is the ratio of number of correct predictions to the total number of the input samples. We generally measured the accuracy with the number of iterations as 50 and achieved a maximum accuracy of 98.13 percent for the training phase. In order to measure the performance of the logistic regression classifier we have used confusion matrix which represents the true positives, false positives, true negatives and false negatives respectively.

The confusion matrix for the logistic regression classifier is as follows:

From the figure 3, we can see that the true negatives are 11375, false positives are 57, false negatives are 177 and true positives are 919.

We have also used the precision as one of the measures which describes the percentage of the results obtained that are relevant. The precision is 94.12 percent which is less compared to random forest classifier, support vector machines and decision tree classifiers.

We have also utilised the recall as the performance measure which is basically the percentage of total relevant results correctly classified by the algorithm. The value achieved for the recall is 83.5 percent. The recall value is greater than the values of all the random forest, decision tree and SVM classifiers.

It is very essential that we maintain a good balance between

```
[[11375    57]
 [  177   919]]
```

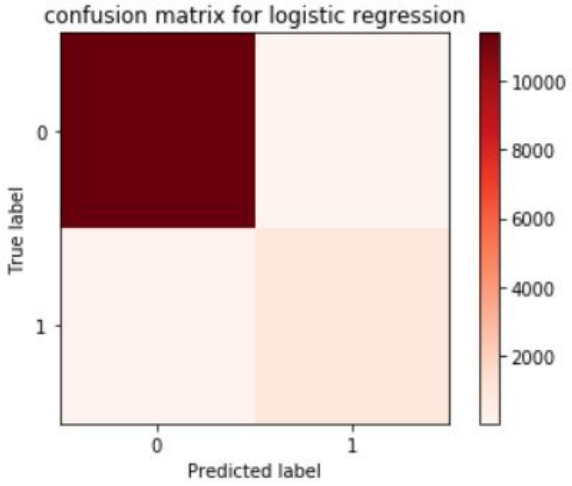

Fig. 3. **Confusion Matrix for Logistic Regression Classifier**

```
true_negative, false_positive, false_negative, true_positive = cm[0][0], cm[0][1], cm[1][0], cm[1][1]
print('true_negative: %d, false_positive: %d, false_negative: %d, true_positive: %d'
      %(true_negative, false_positive, false_negative, true_positive))
```

```
true_negative: 11375, false_positive: 57, false_negative: 177, true_positive: 919
```

Fig. 4. **Values obtained from confusion matrix for Logistic Regression**

precision and recall. Hence, there is an other factor which states the balance called as f1 score. Instead of balancing both the values, it is better that we maximise a single value which is the f1 score. In simple words, the f1 score is the harmonic mean between precision and recall. It is not possible to balance both the quantities at the same time as changes in one might affect the value of the other. Hence, we use f1 score. The f1 score achieved is 88.7 percent. The f1 score value is less when compared to SVM. It is greater when compared to all the other classifiers.

There is also another performance measure we have considered namely AUC-ROC curve(Area under receiver operating statistic curve). The AUC-ROC curve is generally the plot between the true positive rates and false positive rates at different threshold settings used for the performance measurement. AUC determines the degree of measure of seperability and ROC is the probability curve. It is mentioned that, higher the area under the curve higher the accuracy of classifying the data appropriately. The AUC-ROC value for the logistic regression classifier is 91.67 percent. This value is greater than all other classifiers.

The final test accuracy achieved for this logistic regression classifier is 97.31 percent which is pretty good.

B. Random Forest

This classifier generally creates a group of decision trees from a randomly taken subset of the training dataset. This

```
precision = true_positive/(true_positive+false_positive)
print('Precision: %f' %precision)
```

```
Precision: 0.941598
```

Fig. 5. **Precision of logistic regression classifier**

```
f1_score = (2*precision*recall)/(precision+recall)
print('f1 score: %f' %((2*precision*recall)/(precision+recall)))
```

```
f1 score: 0.887066
```

Fig. 6. **Recall value for Logistic Regression Classifier**

would then integrate the number of votes from various decision trees to determine the final target class of a single object. Random forest classifier is basically an ensemble algorithm which combines several other algorithms of same or diverse set of classifying objects. We have also used a variable called as the **n estimators** which decides on the depth of the tree i.e the number of decision trees used. It is basically determined on the basis of accuracy.

*1) **Evaluation of the Random forest results**:* In the same way as above, we consider the classification accuracy as the performance measure in evaluating the results. But during these results the accuracy is not constant and it varies accordingly with the change of the value in the number of decision trees that are used. Also the entropy can also be taken into consideration. We would take the accuracy of that model in which the entropy is very low and the conclusions can be drawn correctly. The training accuracy that is achieved for the model that has less entropy is 98.15 percent during the training phase for a tree depth of 5. To measure the performance of this classifier we have used confusion matrix which represents the true positives, false positives, true negatives and false negatives respectively.

The confusion matrix for the Random forest classifier is as follows:

From the figure 9, we can see that the true negatives are 11382, false positives are 50, false negatives are 181 and true positives are 915.

We have also used the precision as one of the measures which describes the percentage of the results obtained that are relevant.The precision value is 94.81 percent.

We have also utilised the recall value to find the performance measure which is basically the percentage of total relevant results correctly classified by the algorithm. The value achieved for the recall is 83.48 percent.

It is very essential that we maintain a good balance between precision and recall. Hence, there is an other factor which states the balance called as f1 score. Instead of balancing both the values, it is better that we maximise a single value which is the f1 score. In simple words, the f1 score is the harmonic mean between precision and recall. It is not possible to balance both the quantities at the same time as changes in one might

```
f1_score = (2*precision*recall)/(precision+recall)
print('f1 score: %f' %((2*precision*recall)/(precision+recall)))
```

```
f1 score: 0.887066
```

Fig. 7. **Values for f1 score for Logistic regression classifier**

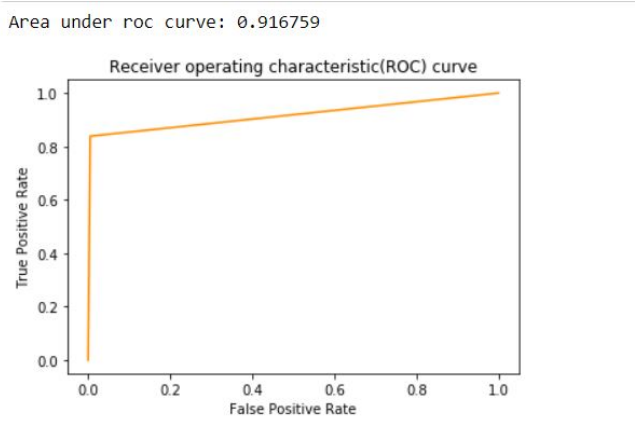

Fig. 8. **AUC-ROC curve for Logistic Regression Classifier**

```
[[11382    50]
 [  181   915]]
```

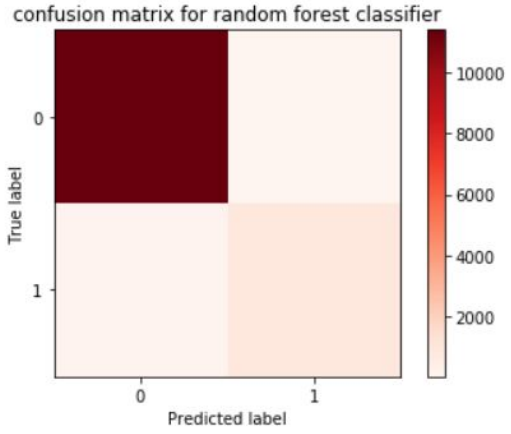

Fig. 9. **Confusion Matrix of Random forest Classifier**

```
true_negative, false_positive, false_negative, true_positive = cm[0][0], cm[0][1], cm[1][0], cm[1][1]
print('true_negative: %d, false_positive: %d, false_negative: %d, true_positive: %d'
      %(true_negative, false_positive, false_negative, true_positive))
```

```
true_negative: 11382, false_positive: 50, false_negative: 181, true_positive: 915
```

Fig. 10. **Values obtained from confusion matrix**

affect the value of the other. Hence, we use f1 score. The f1 score achieved is 88.79 percent.

There is also another performance measure we have considered namely AUC-ROC curve(Area under receiver operating statistic curve). The AUC-ROC curve is generally the plot between the true positive rates and false positive rates at different threshold settings used for the performance measurement. AUC determines the degree of measure of seperability and ROC is the probability curve. It is mentioned that, higher the area under the curve higher the accuracy of classifying the data appropriately. The AUC-ROC value for the Random forest classifier is 91.52 percent.

The final test accuracy achieved for the random forest classifier is 97.80 percent which is pretty good.

C. *Support Vector Machines*

Support Vector Machine is basically a discriminate classifier which can be addressed as separating the hyperplane. When a labeled training data is given as input for the Support vector machine, the algorithm generally outputs an optimal hyperplane where the categorization of new test examples can be done. In two dimensional space, this hyperplane is basically a line dividing the plane in two parts where each class is present on the either side of the plane.

In our project, we are basically making the hyperplane learn from the linear SVM which can be transformed by the usage of the linear algebra. It is used in the support vector machine classification.

*1) **Evaluation of the SVM results**:* In the same way as above, we consider the classification accuracy as the performance measure. The accuracy in this classifier also varies on the basis of number of iterations on the data. The accuracy we have achieved for the 200 iterations is 98.03 percent. In order to measure the performance of the SVM classifier we have used confusion matrix which represents the true positives, false positives, true negatives and false negatives respectively.

The confusion matrix for the SVM classifier is as follows:

From the figure 15, we can see that the true negatives are 11379, false positives are 53, false negatives are 193 and true positives are 903.

We have also used precision as one of the measures which describes the percentage of the results obtained that are relevant. The precision value is 94.45 percent which is less compared to random forest classifier. It is greater than the precision of the logistic regression and decision tree classifiers.

We have also utilised the recall value to find the performance measure which is basically the percentage of total relevant results correctly classified by the algorithm. The value achieved for the recall is 82.39 percent.

It is very essential that we maintain a good balance between precision and recall. Hence, there is an other factor which states the balance called as f1 score. Instead of balancing both the values, it is better that we maximise a single value which is the f1 score. In simple words, the f1 score is the harmonic mean between precision and recall. It is not possible to balance both the quantities at the same time as changes in one might affect the value of the other. Hence, we use f1 score. The f1 score achieved is 88.01 percent.

There is also another performance measure we have considered namely AUC-ROC curve(Area under receiver operating

```
precision = true_positive/(true_positive+false_positive)
print('Precision: %f' %precision)
```

```
Precision: 0.948187
```

Fig. 11. **Precision of Random Forest classifier**

```
recall = true_positive/(true_positive+false_negative)
print('Recall: %f' %recall)
```

```
Recall: 0.834854
```

Fig. 12. **Recall value for Random Forest Classifier**

statistic curve). The AUC-ROC curve is generally the plot between the true positive rates and false positive rates at different threshold settings used for the performance measurement. AUC determines the degree of measure of seperability and ROC is the probability curve. It is mentioned that, higher the area under the curve higher the accuracy of classifying the data appropriately. The AUC-ROC value for the SVM classifier is 90.96 percent.

The final test accuracy achieved for this random forest classifier is 97.16 percent which is pretty good.

D. Decision trees

Decision tree is basically a flowchart-like tree structure in which every internal node represents a feature, the branch represents a decision rule. The output is described by the leaf node. The node which is initialised first or the node on the highest level is called as root node. It basically partitions according to the value of the attribute or feature. The main aspect of this model is to select the appropriate features. To do this we use the functions like the information gain and the gini imputity in order to minimize the loss function and enhance the accuracy of the whole training model. It follows a method of recursive partitioning for the classification of attributes.

*1) **Evaluation of Decision tree results**:* In the same way as above, we would consider the classification accuracy as the performance measure. Here, the accuracy would vary from depth to depth. The best accuracy we have achieved is 98.47 percent with minimum loss. In order to measure the performance of the decision tree classifier we have used confusion matrix which represents the true positives, false positives, true negatives and false negatives respectively.

```
f1_score = (2*precision*recall)/(precision+recall)
print('f1 score: %f' %((2*precision*recall)/(precision+recall)))
```

```
f1 score: 0.887918
```

Fig. 13. **Values for f1 score for Random Forest Classifier**

```
Area under roc curve: 0.915240
```

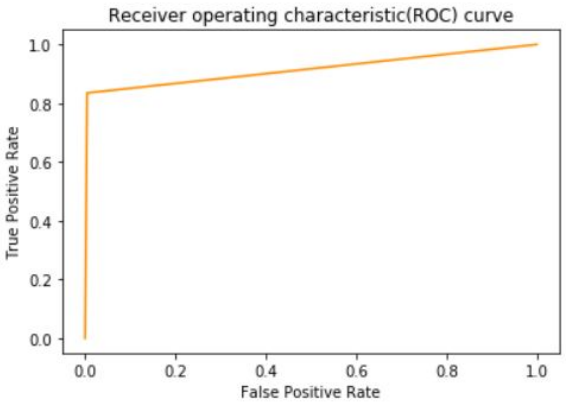

Fig. 14. **AUC-ROC curve for Random Forest Classifier**

```
[[11379    53]
 [  193   903]]
```

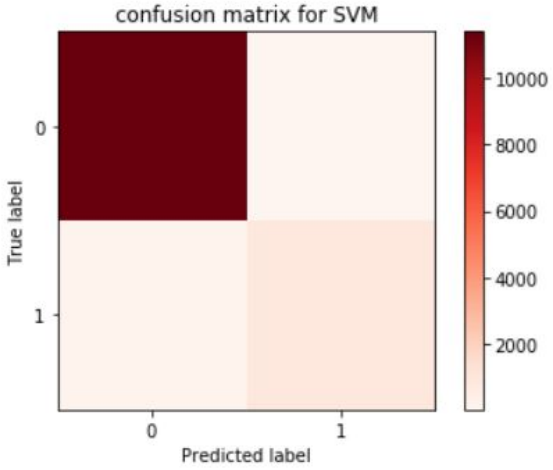

Fig. 15. **Confusion Matrix for SVM Classifier**

```
true_negative, false_positive, false_negative, true_positive = cm[0][0], cm[0][1], cm[1][0], cm[1][1]
print('true_negative: %d, false_positive: %d, false_negative: %d, true_positive: %d'
      %(true_negative, false_positive, false_negative, true_positive))
```

```
true_negative: 11379, false_positive: 53, false_negative: 193, true_positive: 903
```

Fig. 16. **Values obtained from confusion matrix for SVM**

```
precision = true_positive/(true_positive+false_positive)
print('Precision: %f' %precision)
```

```
Precision: 0.944561
```

Fig. 17. **Precision of SVM classifier**

```
recall = true_positive/(true_positive+false_negative)
print('Recall: %f' %recall)
```

```
Recall: 0.823905
```

Fig. 18. **Recall value for SVM Classifier**

```
f1_score = (2*precision*recall)/(precision+recall)
print('f1 score: %f' %((2*precision*recall)/(precision+recall)))
```

```
f1 score: 0.880117
```

Fig. 19. **Values for f1 score for SVM classifier**

The confusion matrix for the decision tree classifier is as follows:

From the figure 21, we can see that the true negatives are 11373, false positives are 59, false negatives are 132 and true positives are 964.

We have also used the precision as one of the measures which describes the percentage of the results obtained that are relevant. The precision value is 94.23 percent which is less compared to random forest classifier and support vector machines. It is greater than the precision of the logistic regression classifier.

We have also utilised the recall value to find the performance measure which is basically the percentage of total relevant results correctly classified by the algorithm. The value achieved for the recall is 87.95 percent. The recall value is greater than the values of all the other classifiers.

It is very essential that we maintain a good balance between precision and recall. Hence, there is an other factor which states the balance called as f1 score. Instead of balancing both the values, it is better that we maximise a single value which is the f1 score. In simple words, the f1 score is the harmonic mean between precision and recall. It is not possible to balance

```
Area under roc curve: 0.909635
```

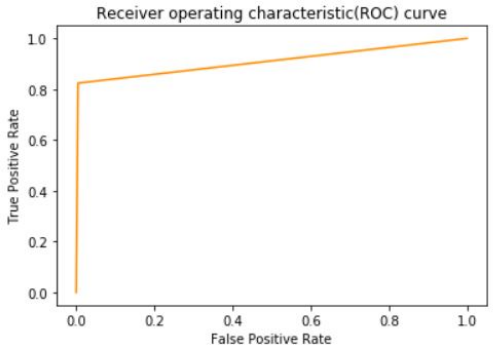

Fig. 20. **AUC-ROC curve for SVM Classifier**

```
[[11373    59]
 [  132   964]]
```

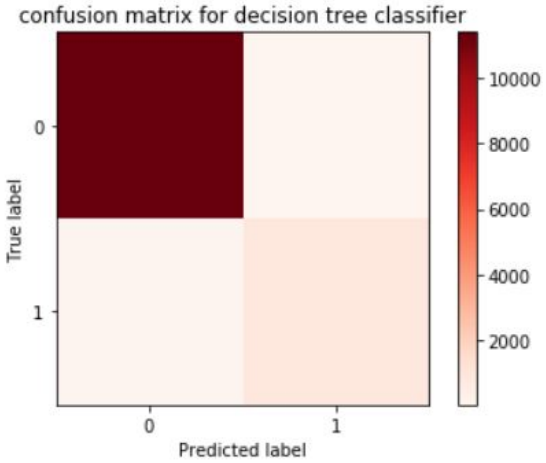

Fig. 21. **Confusion Matrix for decision tree Classifier**

```
true_negative, false_positive, false_negative, true_positive = cm[0][0], cm[0][1], cm[1][0], cm[1][1]
print('true_negative: %d, false_positive: %d, false_negative: %d, true_positive: %d'
      %(true_negative, false_positive, false_negative, true_positive))
```

```
true_negative: 11373, false_positive: 59, false_negative: 132, true_positive: 964
```

Fig. 22. **Values obtained from confusion matrix for decision trees**

both the quantities at the same time as changes in one might affect the value of the other. Hence, we use f1 score. The f1 score achieved is 90.98 percent. The f1 score value is also greater when compared to other classifiers. Hence, the decision tree classifier has the maximum performance measure as well as the train and test accuracy.

There is also another performance measure we have considered namely AUC-ROC curve(Area under receiver operating statistic curve). The AUC-ROC curve is generally the plot between the true positive rates and false positive rates at differ-

```
precision = true_positive/(true_positive+false_positive)
print('Precision: %f' %precision)
```

```
Precision: 0.942326
```

Fig. 23. **Precision of Decision tree classifier**

```
recall = true_positive/(true_positive+false_negative)
print('Recall: %f' %recall)
```

```
Recall: 0.879562
```

Fig. 24. **Recall value for Decision tree Classifier**

```
f1_score = (2*precision*recall)/(precision+recall)
print('f1 score: %f' %((2*precision*recall)/(precision+recall)))
```

```
f1 score: 0.909863
```

Fig. 25. **Values for f1 score for decision tree classifier**

```
Area under roc curve: 0.937201
```

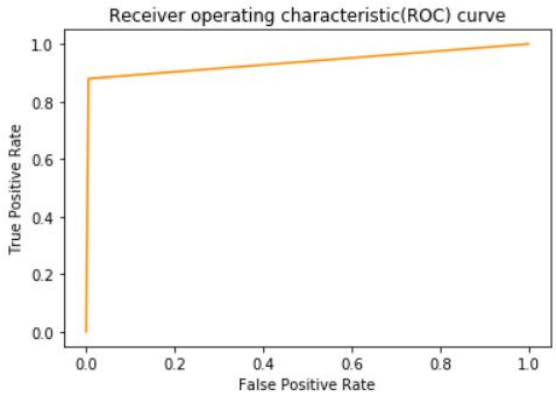

Fig. 26. **AUC-ROC curve for Decision tree Classifier**

ent threshold settings used for the performance measurement. AUC determines the degree of measure of seperability and ROC is the probability curve. It is mentioned that, higher the area under the curve higher the accuracy of classifying the data appropriately. The AUC-ROC value for the Decision Tree classifier is 93.72 percent. This value is greater than that of random forest and SVM classifiers but less than that of logistic regression classifier.

The final testing accuracy achieved for this decision tree classifier is 97.59 percent which is pretty good.

VII. EXPERIMENTS

In our project, we have applied our classifiers in several ways. Initially, we have experimented with the split ratio of training and testing data. We took the split ratio as 0.4 initially but ended up with the ratio of 0.3. We have experimented with two of our classifiers namely logistic regression and support vector machines without importing them from the sci-kit learn library. In those two classifiers the step size, hyperparameter values and the number of iterations were experimented to obtain effective accuracy values. We have achieved almost similar accuracy in both the cases of importing and not importing the library.

VIII. COMPARISON OF RESULTS

Performance measure	RF	DT	LR	SVM
Train accuracy	97.99	98.47	98.13	98.03
Precision	94.8	94.2	94.1	94.4
Recall	83.4	87.9	83.5	82.3
f1 score	88.79	90.9	88.7	88.0
AUC-ROC	91.5	93.7	96.17	90.9
Test accuracy	97.8	97.59	97.31	97.16

IX. CONCLUSION

From the accuracy values, we can understand that random classifier has the better train and test accuracy compared to the other classifiers.

REFERENCES

[1] 1996.http://noiselab.ucsd.edu/ECE228/Reports/Report6.pdf
[2] [**baseline**]https://github.com/praveenachar/pulsar-star-prediction/blob/master/pulsar
[3] [**Dataset**]https://www.kaggle.com/pavanraj159/predicting-a-pulsar-star
[4] https://medium.com/analytics-vidhya/predicting-pulsar-stars-996e22440cf7
[5] https://medium.com/duke-ai-society-blog/classifying-pulsar-stars-using-ai-techniques-d2be70c0f691
[6] https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.htmlsphx-glr-auto-examples-model-selection-plot-roc-py
[7] https://docs.scipy.org/doc/numpy/reference/generated/numpy.sign.html