CS 657 Final Project - Exoplanets in interstellar Space

Vishal Shanmugan Nadar(G01276965)

1. Introduction

The information describes how the flux (light intensity) of thousands of stars changes over time. There is a binary label of 2 or 1 for each star. 2 meant that there was evidence of at least one exoplanet orbiting the star; other sightings are of multi-planet systems. As you might expect, while planets orbit stars that emit light, planets themselves do not. If the star is observed for several months or years, the flux may occasionally "dim" on a regular basis (the light intensity). This is proof that the star might be surrounded by an orbiting body; such a star could be a "candidate" system. Further study of our candidate system, for example by a satellite that captures light at a different wavelength, could solidify the belief that the candidate can be 'confirmed'. Flux Diagram In the above diagram, a star is orbited by a blue planet. At t = 1, the starlight intensity drops because it is partially obscured by the planet, given our position. The starlight rises back to its original value at t = 2. The graph in each box shows the measured flux (light intensity) at each time interval.

The data presented here are cleaned and are derived from observations made by the NASA Kepler space telescope. The Mission is ongoing - for instance data from Campaign 12 was released on 8th March 2017. Over 99% of this dataset originates from Campaign 3. To boost the number of exoplanet-stars in the dataset, confirmed exoplanets from other campaigns were also included. To be clear, all observations from Campaign 3 are included. And in addition to this, confirmed exoplanet-stars from other campaigns are also included. The datasets were prepared late-summer 2016. Campaign 3 was used because 'it was felt' that this Campaign is unlikely to contain any undiscovered (i.e., wrongly labelled) exoplanets. NASA open sources the original Kepler Mission data and it is hosted at the Mikulski Archive. After being beamed down to Earth, NASA applies de-noising algorithms to remove artefacts generated by the telescope. The data - in the. fits format - is stored online. And with the help of a seasoned astrophysicist, anyone with an internet connection can embark on a search to find and retrieve the datafiles from the Archive.

2. Methods and Techniques

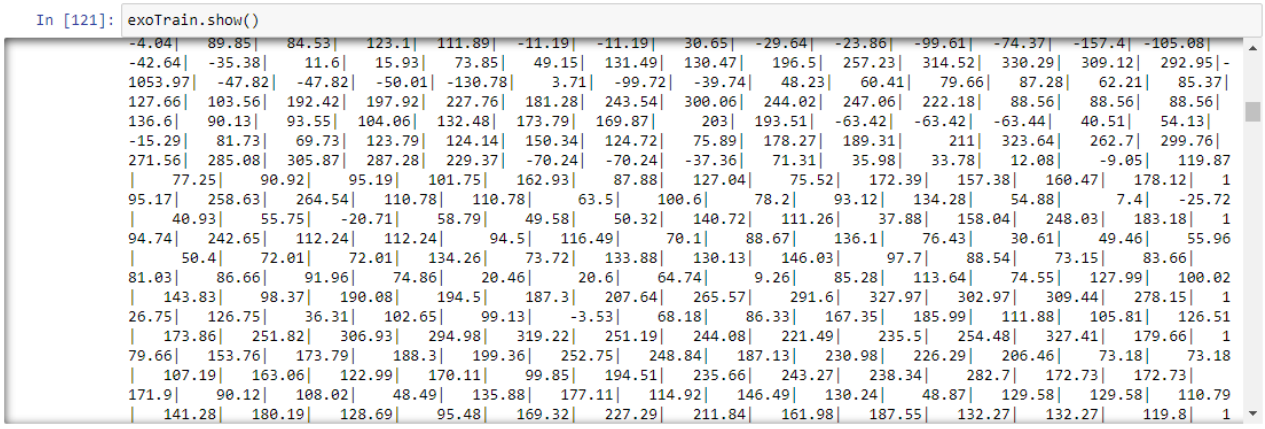
The dataset we are planning to use for this project is available here. This dataset is of about 291 MB in size, and it consists of 3198 features. There are two CSV which are exoTrain.csv and expTest.csv. The exoTrain.csv consists of train data in which there are 5087 rows, 3198 features and label column with values 1 and 2 and 37 confirmed exoplanet-stars and 5050 non-exoplanet-stars. The exoTest.csv consists of test data in which there are 570 rows,3198 features and 5 confirmed exoplanets and 565 non-exoplanet-stars in test data.

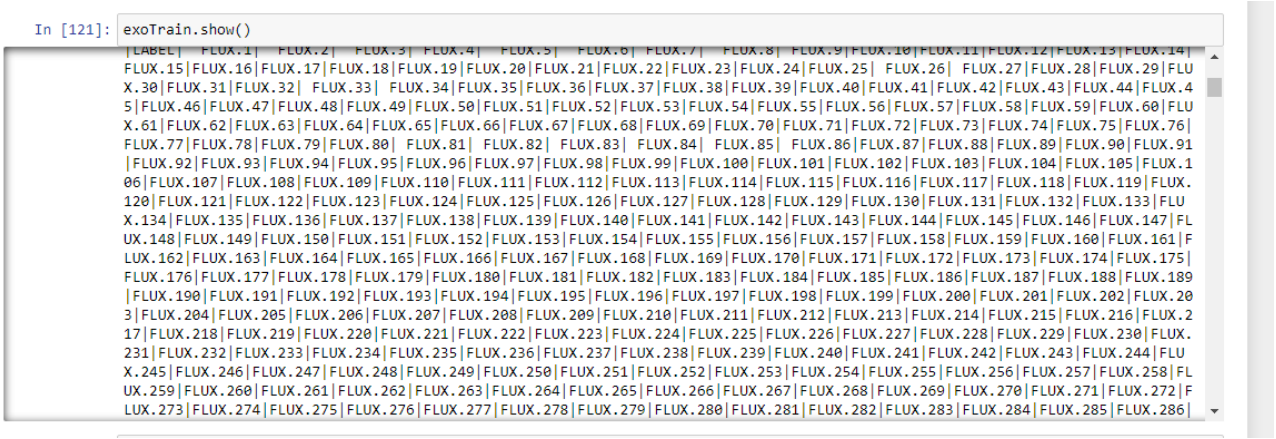
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	FLUX.10	FLUX.11	FLUX.12	FLUX.13	FLUX.14	FLUX.15	FLUX.16	FLUX.17	FLUX.18	FLUX.19	F
2	93.85	83.81	20.1	-26.98	-39.56	-124.71	-135.18	-96.27	-79.89	-160.17	-207.47	-154.88	-173.71	-146.56	-120.26	-102.85	-98.71	-48.42	-86.57	
2	-38.88	-33.83	-58.54	-40.09	-79.31	-72.81	-86.55	-85.33	-83.97	-73.38	-86.51	-74.97	-73.15	-86.13	-76.57	-61.27	-37.23	-48.53	-30.96	
2	532.64	535.92	513.73	496.92	456.45	466	464.5	486.39	436.56	484.39	469.66	462.3	492.23	441.2	483.17	481.28	535.31	554.34	562.8	
2	326.52	347.39	302.35	298.13	317.74	312.7	322.33	311.31	312.42	323.33	311.14	326.19	313.11	313.89	317.96	330.92	341.1	360.58	370.29	
2	-1107.21	-1112.59	-1118.95	-1095.1	-1057.55	-1034.48	-998.34	-1022.71	-989.57	-970.88	-933.3	-889.49	-888.66	-853.95	-800.91	-754.48	-717.24	-649.34	-605.71	
2	211.1	163.57	179.16	187.82	188.46	168.13	203.46	178.65	166.49	139.34	146.76	149.16	162.55	159.35	173.32	108.24	175.35	143.4	119.3	
2	9.34	49.96	33.3	9.63	37.64	20.85	4.54	22.42	10.11	40.1	-13.05	-3.15	-31.81	-0.86	-22.25	-34.21	-15.52	-30.44	-40.06	
2	238.77	262.16	277.8	190.16	180.98	123.27	103.95	50.7	59.91	110.19	16.41	43.09	-47.67	-75.14	-86.19	-31.11	-10.91	29.17	-13.06	
2	-103.54	-118.97	-108.93	-72.25	-61.46	-50.16	-20.61	-12.44	1.48	11.55	38.69	38.82	49.03	56.92	46.09	59.42	174.29	81.03	64.6	
2	-265.91	-318.59	-335.66	-450.47	-453.09	-561.47	-606.03	-712.72	-685.97	-753.97	-682.34	-772.59	-778.41	-760.66	-663.09	-609.03	-545.47	-510.28	-382.03	
2	118.81	110.97	79.53	114.25	48.78	3.12	-4.09	66.2	-26.02	-52.34	-41.88	-93.27	-80.64	-200.33	-224.16	-249.28	-349.98	-353.38	-513.28	
2	-239.88	-164.28	-180.91	-225.69	-90.66	-130.66	-149.75	-120.5	-157	-114.19	-70.66	-120.5	-89.31	-167.47	-130.22	-140.19	-168.91	-158.69	-111.91	
2	70.34	63.86	58.37	69.43	64.18	52.7	47.58	46.89	46	38.88	46.46	28.3	26.09	25.13	14.43	16.7	10.24	24.28	23.92	
2	424.14	407.71	461.59	428.17	412.69	395.58	453.35	410.45	402.09	380.14	393.22	369.66	369.16	416.62	411.73	417.77	414.84	419.27	411.11	
2	-267.21	-239.11	-233.15	-211.84	-191.56	-181.69	-164.77	-156.68	-139.23	-127.77	-110.32	-106.93	-87.59	-84.56	-66.93	-53.98	-45.6	-27.34	-25.89	
2	35.92	45.84	47.99	74.58	87.97	87.97	105.23	131.7	130	155.23	163.73	170.26	176.66	197.1	190.25	185.98	232.13	233.66	238.56	
2	-122.3	-122.3	-131.08	-109.69	-109.69	-95.27	-93.93	-84.84	-73.65	-59.87	-43.83	-47.77	-53.07	-44.01	-38.11	-38.94	-26.95	-0.76	14.55	
2	-65.2	-76.33	-76.23	-72.58	-69.62	-74.51	-69.48	-61.06	-49.29	-56.45	-57.62	-46.35	-43.91	-23.52	-27.9	-16.83	-7.34	-13.62	-5.32	
2	-66.47	-15.5	-44.59	-49.03	-70.16	-85.53	-52.06	-73.41	-59.69	-41.78	-4.78	10.09	8.19	4.62	-25.94	-23.38	0.81	22.09	20.12	

For the methodology, we will be pre-processing the data by converting the features and the label into compactible with ml models and then running different machine learning models to predict the accuracy of the models. The different ml models which will be used are Logistic Regression, Naïve Bayes, Decision Tree Classifier, Gradient Boost Trees and One Vs Rest. I will then also use confusion matrix and roc to define the correctness of the best ml model.

Data Preprocessing:

First, we load both the csv files into spark dataframe using spark read csv. The loaded data initially looks as follows:





After loading the data, I converted the label column from values 1 and 2 to 0 and 1 respectively because some ml models use binary labels for classification. Also, the column names contain “.” Full stop in them which cause issues in spark while reading the name, so I change the full stop(.) in column name to underscore(_) .

I also analyzed the datatype of all columns and found it to be string. I then casted the label column to int as it contained only 1 and 2 valued labels. As the rest of the flux columns contains decimal values, so I have casted the columns as float. The changed data type can be seen in the images below: -

```
exoTrain.printSchema()
```

```
root
 |-- LABEL: string (nullable = true)
 |-- FLUX.1: string (nullable = true)
 |-- FLUX.2: string (nullable = true)
 |-- FLUX.3: string (nullable = true)
 |-- FLUX.4: string (nullable = true)
 |-- FLUX.5: string (nullable = true)
 |-- FLUX.6: string (nullable = true)
 |-- FLUX.7: string (nullable = true)
 |-- FLUX.8: string (nullable = true)
 |-- FLUX.9: string (nullable = true)
 |-- FLUX.10: string (nullable = true)
 |-- FLUX.11: string (nullable = true)
 |-- FLUX.12: string (nullable = true)
 |-- FLUX.13: string (nullable = true)
 |-- FLUX.14: string (nullable = true)
 |-- FLUX.15: string (nullable = true)
 |-- FLUX.16: string (nullable = true)
 |-- FLUX.17: string (nullable = true)
```

```
exoTrain.printSchema()
```

```
root
 |-- LABEL: integer (nullable = true)
 |-- FLUX_1: float (nullable = true)
 |-- FLUX_2: float (nullable = true)
 |-- FLUX_3: float (nullable = true)
 |-- FLUX_4: float (nullable = true)
 |-- FLUX_5: float (nullable = true)
 |-- FLUX_6: float (nullable = true)
 |-- FLUX_7: float (nullable = true)
 |-- FLUX_8: float (nullable = true)
 |-- FLUX_9: float (nullable = true)
 |-- FLUX_10: float (nullable = true)
 |-- FLUX_11: float (nullable = true)
 |-- FLUX_12: float (nullable = true)
 |-- FLUX_13: float (nullable = true)
 |-- FLUX_14: float (nullable = true)
 |-- FLUX_15: float (nullable = true)
 |-- FLUX_16: float (nullable = true)
 |-- FLUX_17: float (nullable = true)
```

Using the isnan function I checked for null values in every column, but no null values were found. After all these steps, I vectorized all the flux columns using VectorAssembler though which I received a single column called features. Thus finally, we select the label column and features column which are used by the ml models for classification.

Building a Machine Learning Model:

After the preprocessing of the data, we moved on to train various machine learning models on our preprocessed dataset. The model which yields the highest accuracy was selected after through testing. The models I used for trained are:

- Logistic Regression
- Naïve Bayes
- Decision Tree Classifier
- Gradient Boost Trees
- One Vs Rest

Following is the description of the models and the hyperparameters used:

Logistic Regression

This method is used to determine the link between the dependent and independent variables and to forecast future events. The parameters being used are maxIter, regParam, elasticNetParam. The best values for the parameters are chosen using hyperparameter tuning and cross-validation. Initially the logisticRegressor was started with maxIter=100. The best parameters are printed out using the bestModel function.

Naive Bayes

Naive Bayes is a simple multiclass classification algorithm with the assumption of independence between every pair of features. The model type used in Naïve Bayes was multinomial. The Naïve Bayes does not take negative values as features, so instead of converting the negative values to positive by adding values to all row, instead MinMaxScaler was applied which turns the features values between 0 to 1. The best parameters are printed out using the bestModel function.

Decision Tree Classifier

Decision Tree builds classifier models in the form of a tree structure. The parameters being used are 'minInstancesPerNode', 'maxDepth', 'maxBins'. The best values for the parameters are chosen using hyperparameter tuning and cross-validation. The best parameters are printed out using the bestModel function.

One Vs Rest

OneVsRest is an example of a machine learning reduction for performing multiclass classification given a base classifier that can perform binary classification efficiently. This model was used with logistic regression, decision tree classifier and gradient Boosted Tress.

Gradient Boost Trees

GBTRegressor trains a large number of decision trees iteratively. The algorithm uses the current ensemble to forecast the label of each training instance on each iteration. The best values for the parameters are chosen using hyperparameter tuning and cross-validation. maxDepth, maxBin and maxIter are the parameters optimized using hyper parameter tuning and cross validation. The best parameters are printed out using the bestModel function.

3. Discussion and Results

3.1 Evaluation Metrics

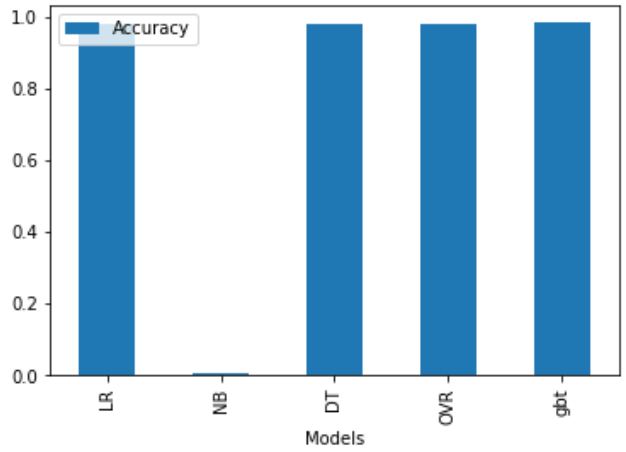
Accuracy is my main evaluation metric for model selection. The dataset already consists of train and test csv for validation purposes so there was no need of splitting data as test and train. I also used cross validator for model evaluation purposes. Finally with the best model, I used confusion matrix and Roc for evaluation purposes.

3.2 Experimental Results

As mentioned above, I have used ParamGridBuilder and CrossValidator to tune the parameters for every algorithm and then checked the scores from our evaluation metrics for every model to finalize the best model. These are the scores for the tuned models:

Model's Name	Accuracy
Linear Regression	0.9789473684210527
Naïve Bayes	0.008771929824561403
Decision Tree Classifier	0.977192982451404
Gradient Boost Classifier	0.9824561403508771
One vs Rest (GBT)	0.9824561403508771

As you can see from the above-mentioned values, Gradient Boost Classifier and One vs Rest were clearly the best model to predict whether the planet is an exoplanet or not.



The above bar graphs shows the accuracy of each model. As you can see the Random Forest classifier and One vs Rest are slightly above in accuracy. The naïve bayes does not perform well as the features values are compressed to range 0 to 1 using MinMaxScaler so the prediction accuracy dropped drastically. After CrossValidator and getting the best parameters for each model the highest accuracy was achieved by most the models of 0.9912280701754386.

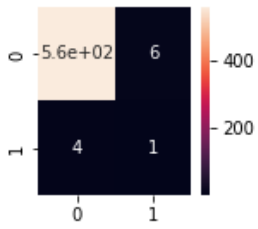
Below are the confusion matrix and its heat map for the Gradient Boost Classifier which is one of the best accuracy classifier for our dataset.

```
print(metrics.confusionMatrix().toArray())
```

```
[[559.   6.]
 [  4.   1.]]
```

```
sn.heatmap(df_cm, annot=True)
```

```
<AxesSubplot:>
```



4. Conclusion

According to the findings, Gradient Boost Classifier and One vs Rest has a better Accuracy score when compared to other machine learning models. When the features values are distributed over large values then MinMaxScaler does not work well as the values are reduced in the range of 0 to 1.Thus, We have created a model to predict whether there is a exo-planet present in the star system with the help of flux intensities.

7. References

[1] Spark ML Classification Models - https://spark.apache.org/docs/latest/ml-classification-regression.html

[2] Classification methods we should know - https://cprosenjit.medium.com/9-classification-methods-from-spark-mllib-we-should-know-c41f555c0425

[3] Regression Models and parameters explanation - https://cprosenjit.medium.com/9-classification-methods-from-spark-mllib-we-should-know-c41f555c0425

[4] Spark by Examples different spark methods implementation - https://github.com/spark-examples/pyspark-examples