PHYS3999

DIRECTED STUDIES IN PHYSICS

UNCOVERING THE NATURE OF FERMI LAT UNASSOCIATED GAMMA-RAY SOURCES

July 25, 2018

Fan Kwok Lung
3035272731

Supervised by:
Dr. Pablo Saz Parkinson
Dr. Stephen Ng

Contents

1 Introduction

1.1 Gamma-Ray astronomy

Gamma Rays are the most energetic electromagnetic waves in the universe.The lower energy limit of gamma-ray photons is $100keV$ and astronomers have detected very-high-energy gamma-ray photons with energy over $10TeV$. It is also known as "non-thermal" electromagnetic radiation due to the fact that it is rarely produced by blackbody radiation since the temperature required to produce enough gamma-ray photons is very high.It is related to a lot of interesting astronomical phenomena. For example, neutron star mergers will create a short gamma-ray burst[1], pulsars and Active Galactic Nuclei will also emit gamma rays[11] .Also, if dark matter is composed of weakly interacting massive particles, it may also annihilate and produce gamma-ray photons[11].Hence gamma-ray astronomy is very important for us to study high energy events and compact astronomical objects. In order to understand these astronomical events and objects better, we need observations in the gamma-ray spectrum. Observations of the gamma-ray photons are done by space-based and ground-based telescope.Ground-based telescopes usually detect the secondary products created by the interaction between hard gamma rays and atmosphere. The flux of such gamma-ray photons($> 100GeV$) is very low [11] .Space-based telescopes like Fermi and DAMPE usually consist of scintillator detectors and semiconductor detectors which can detect mid-level(MeV to TeV) energy gamma rays and cosmic rays. The Fermi Large Area Telescope(LAT) is a gamma-ray space telescope and it has already detected more than 5000 gamma point sources including AGN, pulsars and others like supernova remnants while the previous generation telescope provided only around 300 sources[14].

1.2 Fermi Large Area Telescope

The Fermi satellite was launched into a low earth orbit in 2008 and there are two scientific instruments on board it.The Gamma-ray Burst Monitor (GBM) which is mainly used to detect Gamma Ray Bursts. One of the most famous results of the GBM was the first detection of the EM counterpart of a binary neutron star merger and the GBM team was awarded the Rossi prize for it. The Fermi Large Area Telescope(LAT) is a low earth telescope which was designed to detect gamma-ray photons with an energy from 20 MeV to over 300GeV.It was designed and built by institutes and universities in France, Italy, Japan, Sweden, and the United States.[5]

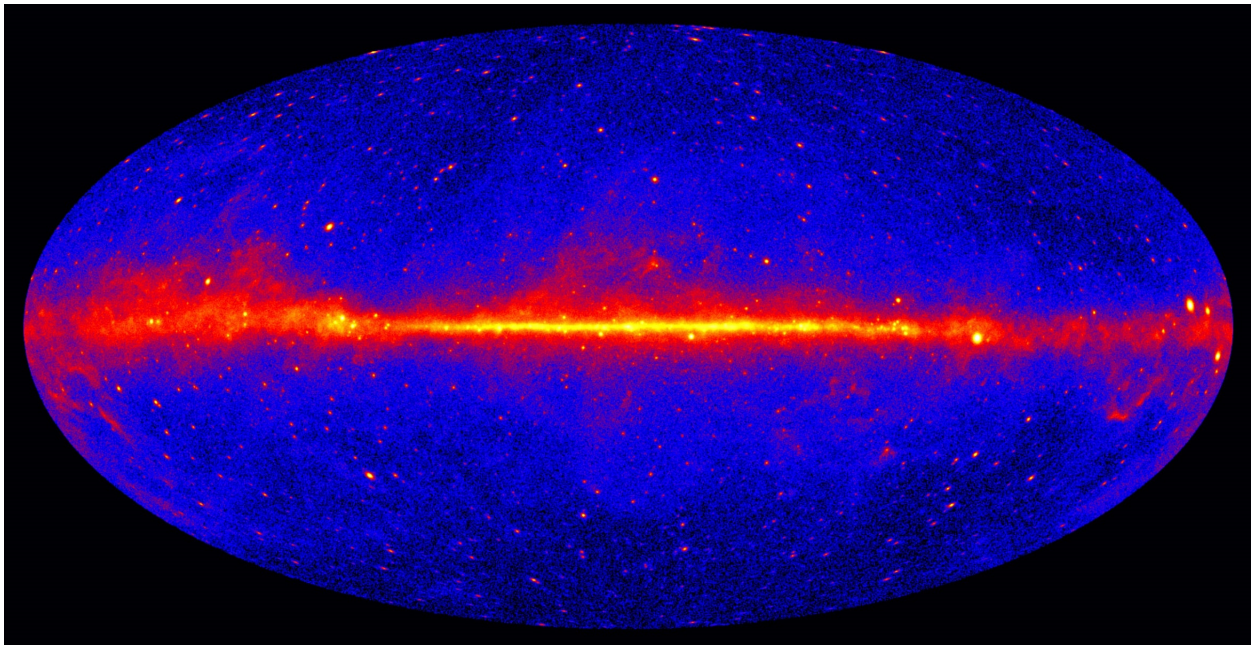

Figure 1: Fermi LAT 5 Year of data(NASA)

1.2.1 Instrument Design

Fermi LAT is a pair-conversion telescope with a precision converter-tracker and calorimeter, each consisting of a 4 × 4 array of 16 modules supported by a low-mass aluminum grid structure [5]. When a gamma ray photon hits the LAT, it will interact with the high-Z material in each layer and create an electron-positron pair and this process is known as pair production.Each layer consists of silicon strip detectors which can detect the charged particle produced by pair production process.Also, LAT consists of a calorimeter which can measure the energy deposition caused by particle shower and image the shower development profile. The calorimeter consists of 96 CsI(Tl) crystals, and each crystal has the size of 2.7 cm × 2.0 cm × 32.6 cm.[5]
Fermi LAT has a large FOV, board energy range , and high point source sensitivity. Over 5000 point sources have been detected by Fermi LAT.

1.2.2 Fermi Source Catalogs

Fermi Large Area Telescope source catalogs contains information on gamma-ray sources which were discovered with the Fermi LAT data. The LAT collaboration has released 0FGL,1FLG,2FLG,3FGL source catalog based on first 3 months,one year,two years and four years of data[2][17][3][4].The detail of catalogs is given in the table below. The last published catalog is the Fermi Large Area Telescope Third Source Catalog(3FGL) which was released in 2015. It used the first four years of Fermi LAT data. The catalog contains 3033 sources with significance larger than

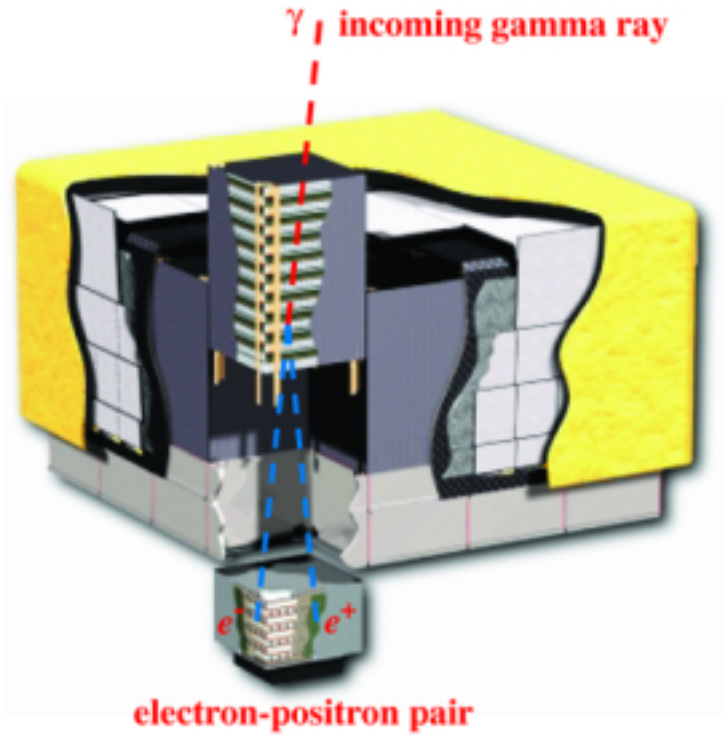

Figure 2: Schematic diagram of the Large Area Telescope

Table 1: Summary of the catalogs

Name of the catalog	Release date	Data used	Number of sources	Unassociated sources
0FGL	February 2009	3 months	205	38
1FGL	May 2010	11 months	1451	630
2FGL	April 2012	2 years	1873	575
3FGL	Jan 2015	4 years	3033	1010
FL8Y	Jan 2018	8 years	5524	2130

4σ. The catalog includes the position of the source as well as other information related to the source[4]. Fermi LAT had now in obit for almost ten years and the fourth Fermi LAT catalog based on 8 years of LAT data will be released very soon. A preliminary LAT 8-year Point Source List(FL8Y) was released in 2018 for scientific research.The official 4FGL Catalog will be released and replace FL8Y.

1.3 Unassociated gamma-ray sources in the catalog

In the Fermi LAT point source catalog, a large portion of the sources are not associated with any known astronomical object. These point sources usually are Active Galactic Nuclei (AGN) or pulsars (PSR) [18].

1.3.1 Active Galactic Nuclei

Active Galactic Nuclei(AGN) are galaxies with a very high luminosity in the central region.The current model of AGN is the accretion materials around the black hole fall into the black hole and the gravitation energy turned in EM energy and radiated out. Also, there will be jets in the two ends of the AGN[15]. There are also some subcategories of AGN like blazar. The AGN detected by Fermi are usually blazars with the jet pointing towards us. Blazars usually have higher variability and the spectrum can be well modeled by a simple power law. The difference between the gamma-ray emission of pulsars and AGN can potentially help us distinguish them.

1.3.2 Pulsars

When a main sequence star has evolved to the end stage and its remnant has a mass higher than 1.4 solar masses but lower than 3.2 solar masses[9], it will further collapse into a neutron star. If a neutron star is fast rotating and it has a very strong magnetic field,it will accelerate electrons and positrons and produce photons which are detectable by telescopes[6].Usually, a pulsar is detected by the radio signal.However, pulsars also emit radiation across the electromagnetic spectrum and with todays sensitive satellites, we can also detect pulsars in the x-ray spectrum and gamma ray spectrum.
The mechanism of gamma-ray emission of pulsars is very complicated and still not fully understood by astronomers.Several models have been proposed to explain the high energy emission of pulsars. Although we still do not completely understand the emission mechanism of pulsars, there are some properties that we know.For example, pulsars have a low variability. Gamma-ray pulsars emit the gamma-ray beams which can be detected if the beam is pointing toward us. There are about 200 known gamma-ray pulsars and some of them are radio quiet.The reason why some pulsars cannot be seen in radio is that the radio beam is emitted at different angle and is probably more narrower[16]. Researchers are trying to identify pulsars at different wavelengths(radio, X-ray, and gamma-ray) and different instruments.

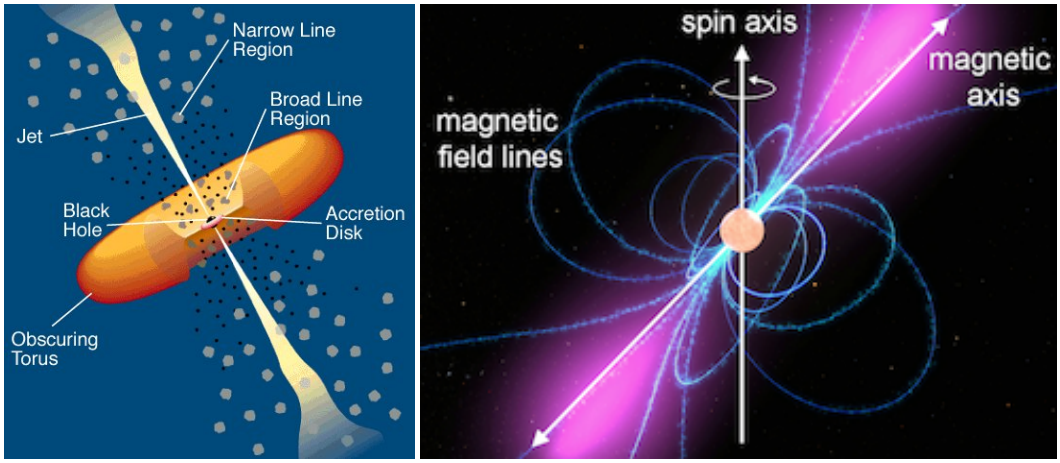

Figure 3: The physical model of active galactic nuclei(left) and pulsars (right)(NASA)

2 Machine learning for source classification

To classify the nature of gamma-ray sources, astronomers usually rely on multi-wavelength analysis and visual analysis.Also, blind searches with only gamma-ray data have successfully detected radio-quiet gamma-ray pulsars[19]. However, these methods can be time-consuming and the visual analysis is not rigorous.Also, since the telescope time and computational time are both very expensive, looking up all gamma-ray sources is not a feasible solution. We want to find an automated way to classify these point sources. Parkinson et al.[18] tried a new approach to classify the unassociated gamma-ray sources.They used logistic regression and Random forest algorithm to train a classification model and predict the nature of the unassociated 3FGL sources. We tried several machine learning algorithms for classification of point sources. These machine learning methods may also give us new understanding about pulsars by revealing the importance of features.

2.1 logistic regression

Logistic regression is a very powerful algorithm that was developed by Cox (1958)[10].It is very useful for binary classification problems.Logistic regression models the probability as:

$$Pr(y=1|X=\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{b}+\mathbf{W}^{\mathsf{T}}\mathbf{x})}} \tag{1}$$

$$Pr(y=0|X=\mathbf{x}) = \frac{e^{-(\mathbf{b}+\mathbf{W}^{\mathsf{T}}\mathbf{x})}}{1+e^{-(\mathbf{b}+\mathbf{W}^{\mathsf{T}}\mathbf{x})}} \tag{2}$$

where $\mathbf{x}, \mathbf{b}, \mathbf{W}$ is an n dimensional vector. We call $\mathbf{W}$ the weight, $\mathbf{b}$ the bias. Using optimization algorithms like gradient descent, we can obtain the value for weight and the bias.Then for each input vector $\mathbf{x}$, we can output the probability of it being class 1 and class 0.

2.2 Random Forests

Random Forests [7] is a type of tree ensemble method.It is similar to the bagging of the decision tree. It consists of a large number of separately built decision trees with randomly selected features and data.With a large number of the uncorrelated decision trees, the model can minimize the generalized error and seldom suffers from the over-fitting problem of the classifier.Due to its robustness, the Random Forest classifier is widely used in different areas. The decision tree is built using usual decision tree algorithm but with only m feature the original feature from the training set during each split where $m < n$ and n is the total number of feature [13].The split will not always be based on the strongest predictor parameter but some other potential strong parameters hence making the classifier more robust[18].

2.3 Support vector machine

Support vector machine is a very powerful classifier which was considered to be the best machine learning algorithm for a long time. The key idea of support vector machine is to try to maximize the functional margin $\hat{j}$ [13].

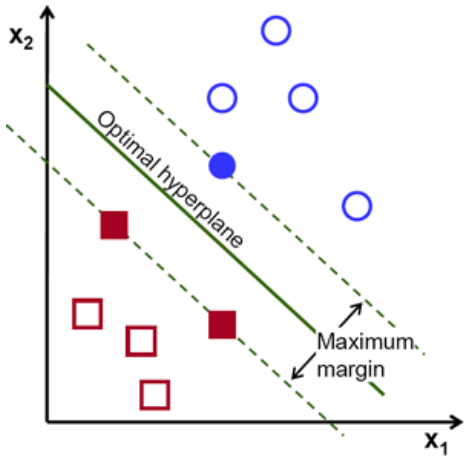

Figure 4: The idea of support vector machine

We define the functional margin as $\hat{j} = \min_i y^{(i)}(\mathbf{W}^\top \mathbf{x}^{(\mathbf{i})} + b)$, where y=1 or -1 ,each representing a class. We aim to find the weight and bias which can minimize the normalized

function margin. Hence the problem is

$$\max_{\mathbf{W},\mathbf{b}} \frac{\hat{j}}{\| \mathbf{W} \|} \tag{3}$$

$$s.t. y^{(i)}(\mathbf{W}^\mathsf{T}\mathbf{x^{(i)}} + b) \geq \hat{j} \tag{4}$$

$$\tag{5}$$

Which is equivalent to

$$\max_{\mathbf{W},\mathbf{b}} \frac{1}{2} \| \mathbf{W} \|^2 \tag{6}$$

$$s.t. y^{(i)}(\mathbf{W}^\mathsf{T}\mathbf{x^{(i)}} + b) \geq 1 \tag{7}$$

Directly tackling this problem is difficult, so we will work with it Lagrangian dual problem:

$$\max \sum_i^n \alpha_i - \frac{1}{2} \sum_i^n \sum_j^n \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i}, \mathbf{x_j} \rangle \tag{8}$$

$$s.t. 0 \leq \alpha_i \forall i \tag{9}$$

$$\sum_i^n \alpha_i y_i = 0 \tag{10}$$

Also, we can add a penalty term ξ_i to tolerate some outliner which may cause the data not to be separated by a linear hyperplane. For non-linear separable data, we can apply kernel trick which replaces the inner product by some polynomial term or Gaussian kernel.

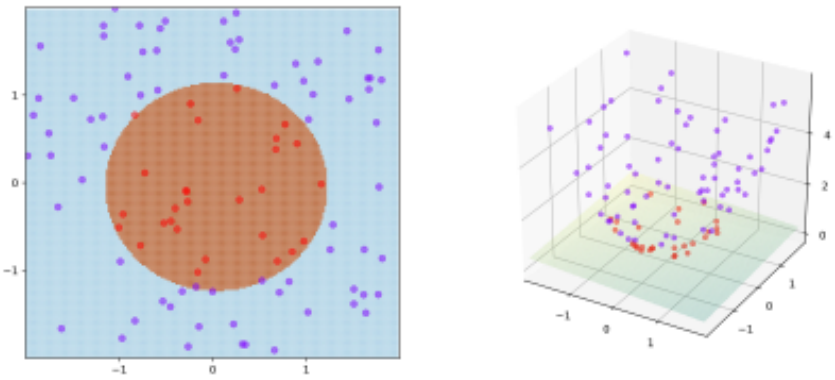

Figure 5: The idea of Kernel trick

2.4 Neural network

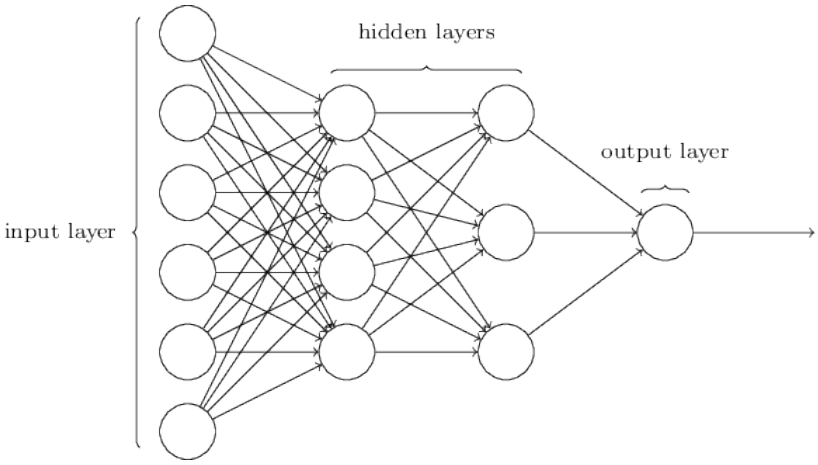

Figure 6: Structure of neural network

Neural networks are well-known algorithm which is widely used in many areas. Although the theoretical explanation of deep neural network is unknown, the performance of neural networks is remarkable.A neural network is made of a multi-layer of neurons with each neuron consisting of a weight vector, bias vector, and an activation function. The weight and bias will be updated during each iteration. The optimization algorithm tries to minimize the cost generated by feeding the data forward.One of the most widely used optimization algorithms is gradient descent. The error of l layer which is defined as $\frac{\partial C}{\partial z^l} = \delta^l$ and the gradient of weight and bias of the l layer and j neuron is just $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ and $\frac{\partial C}{\partial b^l_j} = \delta^l_j$,where a^l is the output of the l layer and k is the kth element of the weight vector. The δ^l can be computed by $((w^{l+1})^\intercal \delta^{l+1})^\intercal \sigma'(z^l)$. By the above formulas, we can calculate the error

of each layer and calculate the gradient of weight and bias of each neuron. This process is called backpropagation and each update is called an iteration [13].

3 Methodology

We will use the data from the preliminary LAT 8-year Point Source List (FL8Y) which was released by the Fermi LAT collaboration in January 2018. We applied several machine learning algorithms including support vector machines, neural networks, random forests and logistic regression. We will check the performance of each algorithm and then try to predict the nature of the unassociated sources.

3.1 Preliminary LAT 8-year Point Source List (FL8Y)

The preliminary LAT 8-year Point Source List[1] was released by the Fermi LAT collaboration for temporary use and will be replaced by 4FGL. Fermi LAT collaboration used the first eight years of LAT science data with the energy range from $100 MeV$ to $1 TeV$ to make the list. The list includes 5524 sources which have a significance greater than 4-sigma, of which 2900 are active galaxies, 217 are pulsars and 2130 sources are unassociated sources.

3.2 Feature selection

We use all the features available and with no missing values to test all the algorithms first and set it as a baseline.Then, we try different subsets of features based on the result. Unfortunately, FL8Y did not include the variability index and it is one of the most important features based on the result of the previous work[18]. The variability of the source is crucial to indicate whether a source is an AGN or a pulsar. Therefore we implemented a method to get an equivalent indicator for variability.

3.3 Variability of the source

Fermi science tools provide two ways to generate light curves of gamma-ray sources, likelihood analysis, and aperture photometry.Likelihood analysis is a more rigorous approach since it leads to a more accurate flux measurement by modeling the background flux.Likelihood analysis leads to greater sensitivity and it is used in the official catalog. However, there are two major drawbacks to likelihood analysis.First is the computational complexity. A likelihood analysis for a source using 8 years of data require hours to complete using a single core, so the time needed for all the sources is enormous. Also, likelihood analysis requires sufficient number of photon in each time bin.FL8Y sources is very faint, so likelihood analysis is not a practical solution.
We will use aperture photometry for the variability analysis. Aperture photometry is basically counting how many photons are detected within a given region of interest and time bin,

[1] https://fermi.gsfc.nasa.gov/ssc/data/access/lat/fl8y/

so the computational complexity is much smaller compared to likelihood analysis. Also, it allows us to use a shorter time bin.
The Perl script we use is modified from the contributed script available from the FSSC page[2] which uses Fermi Science Tools to calculate the photon number and exposure time.
Using aperture photometry, each source requires approximately 2.5 hours on a 2.53GHz core for computing the 8-years light curve. We use the gridpoint system supported by ITS, HKU for the light curve generation.
After the generation of the light curve, we still need a way to generate an index that can indicate the variability of the source. We tried several methods of which gave similar performance is insignificant.Hence we applied the simplest way to generate a variability is using the variance of the flux.We used 30-days time bins to do our analysis.The formula for variability index is

$$V = \sum_{i=1}^{n} \frac{(F_i - \overline{F})^2}{n} \tag{11}$$

where v is the variability index, F_i and $\overline{F}$ is the flux per second per square centimeter of the time bin i and the average flux.

3.4 Imbalance class problem

In our classification tasks, the most difficult problem that we need to tackle is the imbalance class problem.Imbalance class problem means in our training set, the number of one class is significantly greater than the other. In our training set, the number of AGN is much greater than the number of pulsars. The problem can cause our classifier to predict the source to be AGN, resulting in a high overall accuracy and AGN accuracy but very low accuracy in the pulsar prediction. This means our model is a biased classifier. We tried to solve the problem in two ways.

3.4.1 K-fold Cross-validation

K-fold Cross-validation is a common way to evaluate the performance and generalization ability of a model. A commonly chosen value of K is 10. K-fold Cross-validation algorithm will first divide the data set into K subsets, and then train the model using $K-1$ subsets, then make a prediction on the remaining subset. After doing it for all K subsets, we will have a prediction result for all the data in the dataset. Then we can compare them with the true value to get the performance of the model.

[2] https://fermi.gsfc.nasa.gov/ssc/data/analysis/user/

Also, we can use the result of cross-validation to tackle the imbalance class problem. If we train our model using an imbalance dataset, the resulting model will tend to predict the source to be the class that has a large portion, which means our classifier will output a probability of pulsar lower than 0.5 even though the source is actually a pulsar. The solution is to tune the threshold probability to be less than 0.5. For example, if the threshold probability is 0.4, some sources with a pulsar probability of 0.45 will be classified to be pulsar while 0.5 thresholds will predict it as AGN.
The problem is how to determine the value and the criteria of a good threshold. We can use the cross-validation result and receiver operating characteristic (ROC) curve to determine the optimal threshold.In the ROC curve graph, the horizontal axis is the false positive rate and the vertical axis is the true positive rate. Hence the curve will pass through (0,0) and (1,1), we can define the optimal value of threshold to be the point where the value of false positive rate minus true positive rate is the minimum.

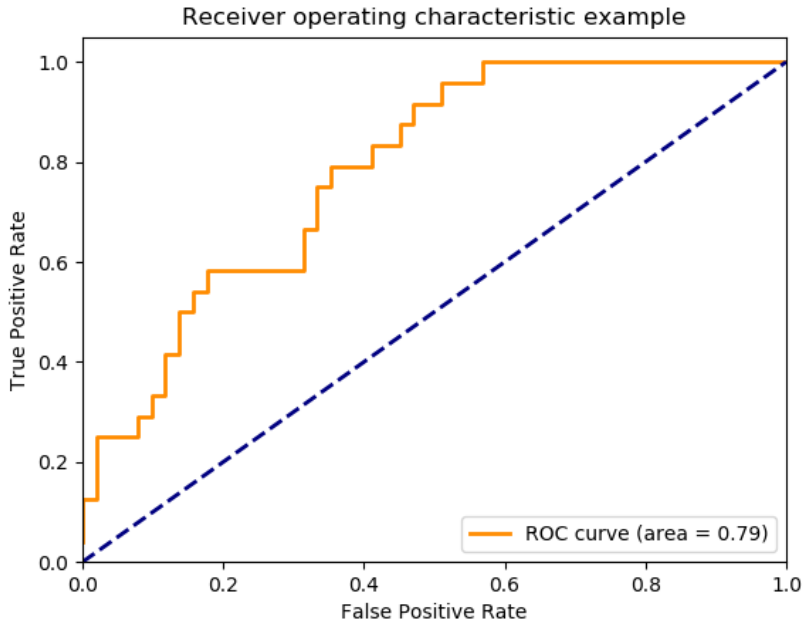

Figure 7: Example of receiver operating characteristic (ROC) curve(sci-kit learn)

3.4.2 Class weight method

In the classification problem, we aim to find a set of parameters such that the loss function or objective function is minimized.Usually, the loss function of a batch of data is calculated by:

$$F_{totalloss} = \frac{1}{N}\sum_{i=1}^{N} F^{(i)}_{individualloss} \tag{12}$$

While the individual loss can be a mean square error function, cross entropy function or other functions. However, the problem with this loss function is that all the class will have an equal weight. This means predicting most of the source to be the class which has the large portion of the training data set can also make the loss go down.But this is biased. A simple modification to the loss function can avoid the problem easily:

$$F_{totalloss} = \frac{1}{N}\sum_{i=1}^{N} W(y^{(i)})F^{(i)}_{individualloss} \tag{13}$$

where W is the weight which depends on the class label of that sample. If we set the weight of the fewer class(in our case is pulsar) to be higher, then it will have a larger effect on the loss function and cause our classify to care more about the pulsar sample during training.A good and simple weight function is

$$W(y^{(i)}) = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(y^{(i)} = 1), & \text{if } y^{(i)} = 1 \\ \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(y^{(i)} = 0), & \text{if } y^{(i)} = 0 \end{cases} \tag{14}$$

3.4.3 Ensemble of two methods

We will use both methods together. In principle, one method is enough for tackling the imbalance class problem, but using two methods should potentially increase the robustness of the model and monitor the model in case something goes wrong.The best threshold should be must closer to 0.5 if we use both methods together.

4 Results

We test the accuracy of the different algorithms using the known association sources. We split the known association dataset into two subsets with 3/7 portion, 70% for training and 30% for testing.We use logistic regression classifier, 500 trees with bootstrap feature Random Forest, (15,9,6,1) neurons ANN(we did not use cross-validation on ANN) and linear kernel support vector machine. The parameter is adjusted based on the result of the accuracy and F-score. The detailed parameter setting can be found in the python script attached.The dependencies of the script include numpy, sci-kit learn, tensorflow and keras.
We had tried different subset of features and found the the following subset of features gave the highest accuracy:Signif_Avg, Npred, Pivot_Energy, Flux_Density, Unc_Flux_Density, Flux100, Unc_Flux100, Flux1000, Unc_Flux1000, Energy_Flux100, Unc_Energy_Flux100, Energy_Flux1000, Unc_Energy_Flux1000, PL_Index, Unc_PL_Index, PLEC_Index, LP_Index, Unc_LP_Index, LP_TSCurv, PLEC_TSCurv, PLEC_Expfactor, Unc_PLEC_Expfactor, Variability_Index.Also, we scale the Flux_Density, Unc_Flux_Density, Flux100, Unc_Flux100, Flux1000, Unc_Flux1000, Energy_Flux100, Unc_Energy_Flux100, Energy_Flux1000, Unc _Energy_Flux1000 by taking the log10. Then, for logistic regression, neural network and support vector machine, we further scale by reducing the feature to zero mean and unit variance since they are sensitive to the feature's numeric magnitude.This is the method that has highest accuracy and consistency.

4.1 Accuracy

The following result is using 0-19 as the random seed for data shuffling and the accuracy for each of the random and also the average accuracy .

From the result above, we can see that the random seed which determines how we split the data set actually affect on the result accuracy, especially on pulsar accuracy.This may due to the imbalance class problem where some random seed split the dataset unevenly such that the proportion of pulsar and AGN differ by a lot in training set and test set.
Using a large number of number(20) of random seed for testing, we should able to get a stable average result on the accuracy.
Average overall accuracy is over 95.9% for all four methods, which shows that the data indeed have an underlying pattern for classification. The pulsar accuracy is slightly below the overall accuracy, all four methods have over 90.4% pulsar accuracy.Random forest has the highest pulsar accuracy which is over 94.8%.And Random forest has the highest overall accuracy which is over 96.2% The high accuracy result show the robustness and reliability of our algorithms.

Table 2: Accuracy of different algorithms with different random seed

RF_Overall	RF_PSR	RF_AGN	LR_Overall	LR_PSR	LR_AGN	ANN_Overall	ANN_PSR	ANN_AGN	SVM_Overall	SVM_PSR	SVM_AGN
0.975	0.953	0.977	0.967	0.922	0.970	0.968	0.938	0.970	0.960	0.906	0.963
0.970	0.889	0.976	0.942	0.903	0.945	0.951	0.931	0.953	0.903	0.944	0.900
0.968	0.935	0.970	0.974	0.887	0.980	0.964	0.952	0.964	0.915	0.903	0.916
0.963	0.971	0.962	0.971	0.838	0.981	0.956	0.926	0.959	0.936	0.941	0.935
0.961	0.910	0.964	0.969	0.821	0.980	0.967	0.881	0.973	0.957	0.881	0.963
0.960	0.972	0.959	0.962	0.915	0.965	0.961	0.958	0.961	0.947	0.958	0.946
0.955	0.952	0.956	0.966	0.905	0.970	0.953	0.968	0.952	0.889	0.968	0.883
0.953	0.962	0.952	0.948	0.937	0.949	0.964	0.949	0.965	0.930	0.924	0.931
0.969	0.975	0.968	0.953	0.938	0.955	0.957	0.975	0.956	0.946	0.938	0.947
0.952	0.914	0.955	0.971	0.871	0.979	0.967	0.900	0.972	0.955	0.929	0.957
0.948	0.951	0.948	0.954	0.918	0.957	0.958	0.934	0.960	0.933	0.918	0.933
0.967	0.942	0.969	0.956	0.907	0.961	0.955	0.919	0.959	0.931	0.919	0.933
0.978	0.939	0.981	0.963	0.924	0.965	0.967	0.955	0.968	0.968	0.970	0.968
0.950	0.965	0.949	0.955	0.906	0.960	0.958	0.906	0.964	0.919	0.953	0.916
0.954	0.984	0.952	0.955	0.935	0.957	0.957	0.952	0.958	0.927	0.968	0.925
0.965	0.942	0.966	0.951	0.928	0.953	0.957	0.942	0.959	0.964	0.826	0.974
0.966	0.955	0.967	0.965	0.939	0.967	0.966	0.939	0.968	0.951	0.939	0.952
0.961	0.942	0.962	0.960	0.913	0.963	0.965	0.928	0.968	0.968	0.841	0.978
0.974	0.945	0.976	0.948	0.945	0.948	0.953	0.945	0.954	0.892	0.945	0.888
0.958	0.970	0.958	0.965	0.909	0.969	0.966	0.924	0.969	0.871	0.970	0.864

Table 3: Average accuracy

RF_Overall	RF_PSR	RF_AGN	LR_Overall	LR_PSR	LR_AGN	ANN_Overall	ANN_PSR	ANN_AGN	SVM_Overall	SVM_PSR	SVM_AGN
0.962	0.948	0.963	0.960	0.908	0.964	0.961	0.936	0.962	0.958	0.904	0.962
0.00835	0.0230	0.00958	0.00857	0.0317	0.0107	0.00517	0.022484795	0.00625	0.0272	0.0392	0.0309

4.2 Consistency

When tackling a classification problem, one important thing is to check the consistency between different algorithms.We will use all known associated sources in the catalog which are AGN or pulsar as the training set and train the models with different methods independently. Then predict whether a source is an AGN or a pulsar in the unassociated sources list with the models trained. If the sources indeed have some underlying pattern, we should get the highly consistent result, all four methods should predict a source as AGN or pulsar simultaneously for most of the time.

We apply the above procedure the test our methods. All the settings and parameters are exactly the same as the previous part. The only difference is now the training set is the whole known associated sources which are AGN or pulsar. Also, instead of evaluating the accuracy with known associated sources, we apply the resulting models to the unassociated sources and see the result of the prediction.

FL8Y includes a total of 2130 unassociated sources, logistic regression predicts 896 of the sources are pulsars, Random Forest predicts 970, neural network predicts 981 and support vector machine predict 980. Almost certainly the true number of pulsars will not be that high but this method will give us a balance between the true positive rate and false negative rate. The overall consistency is 85% which mean 85% of the time, all four methods predict the sources to be AGN or Pulsar together. This means we are confident to say that those 1810

sources really belong to AGN class or Pulsar class as we predicted.Notice that each method is trained independently, which means that the high consistency suggests the high reliability of our model.
Also, we can compare the consistency between only two models. The two best-performing algorithms, Random Forest model and neural network model have 93% of consistency. It is worth nothing that is the logistic regression and support vector machine pair actually have the highest consistency rate,at 94.6%. This is expected since we use a linear kernel for support vector machine, so both classifiers should have a linear decision boundary and it should behave similarly when we use the same training set to train them.

5 Visualization and physical interpretation

After evaluating the model by testing it on the data and checking its accuracy and consistency, we would like to visualize the data and try to find the physical meaning in the data.

5.1 Importance of individual features

Since we have used a large number of features to train our model, we cannot directly visualize such high dimensional data. We first try to evaluate the importance of each feature. Also, by evaluating the feature importance, we can also get an intuitive idea of why these features can predict whether the source is an AGN or a pulsar.
Random Forest provides a way to evaluate the feature importance. We use the mean decrease in impurity to measure the importance of features.Mean decrease in impurity of a feature is defined as the average decrease in Gini index when the node is split base on that feature. A high decrease in Gini index means the feature play a very important role in the classification since it can split the classes significantly.

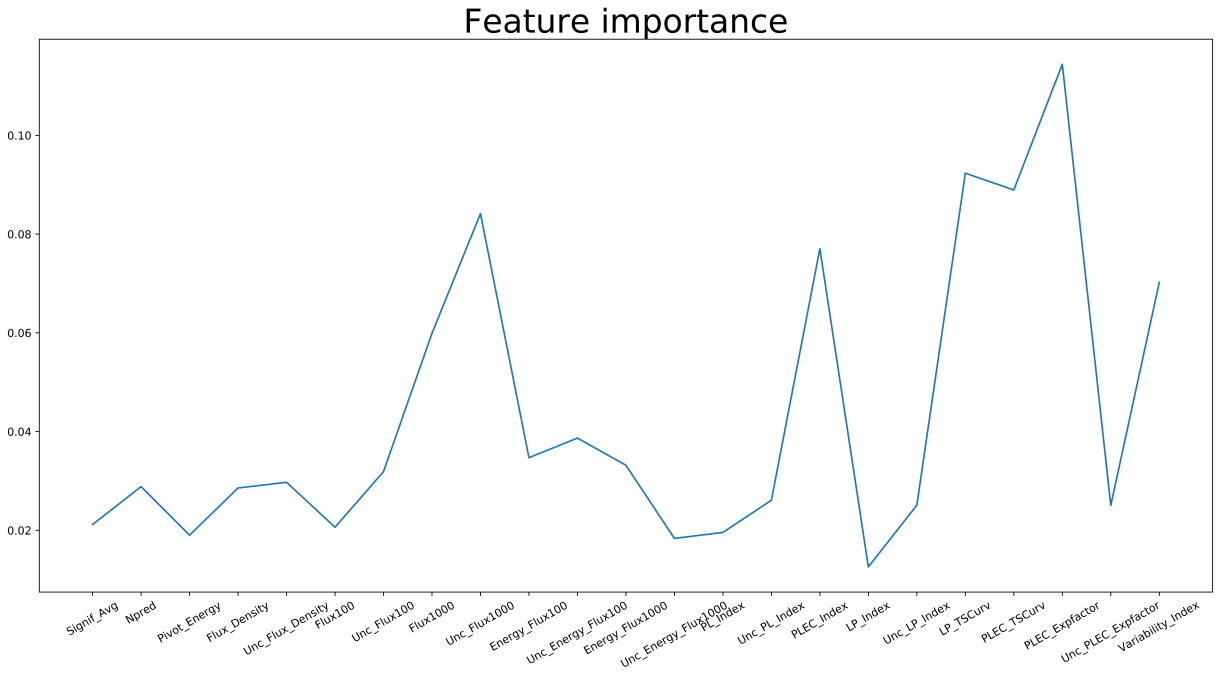

Figure 8: The mean decrease in impurity(sci-kit learn)

In fig.8, we can see that Unc_Flux1000, PLEC_Index, LP_TSCurv, PLEC_TSCurv, PLEC_Expfactor and Variability_Index are the important features. This agree with our intuition which the variability of the source, curvature and spectral index of energy spectrum indicating the nature of the source. We further show the one variable correlation with the class.

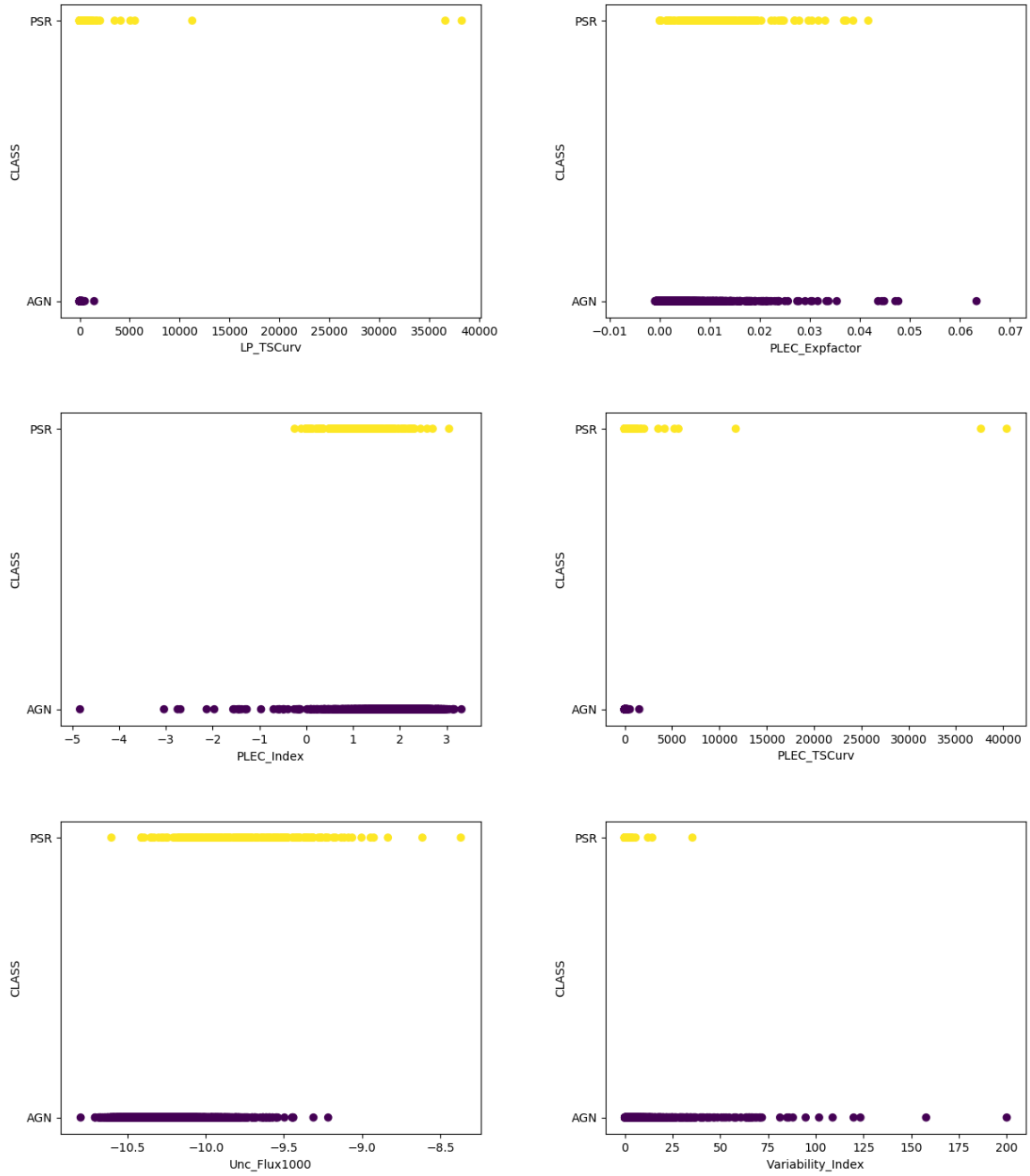

Figure 9: The PSR and AGN class as y-axis, the feature as x axis

From fig.9, we can see that these important features indeed have the implication of whether the source is an AGN or PSR. We further examine the two variable figures.

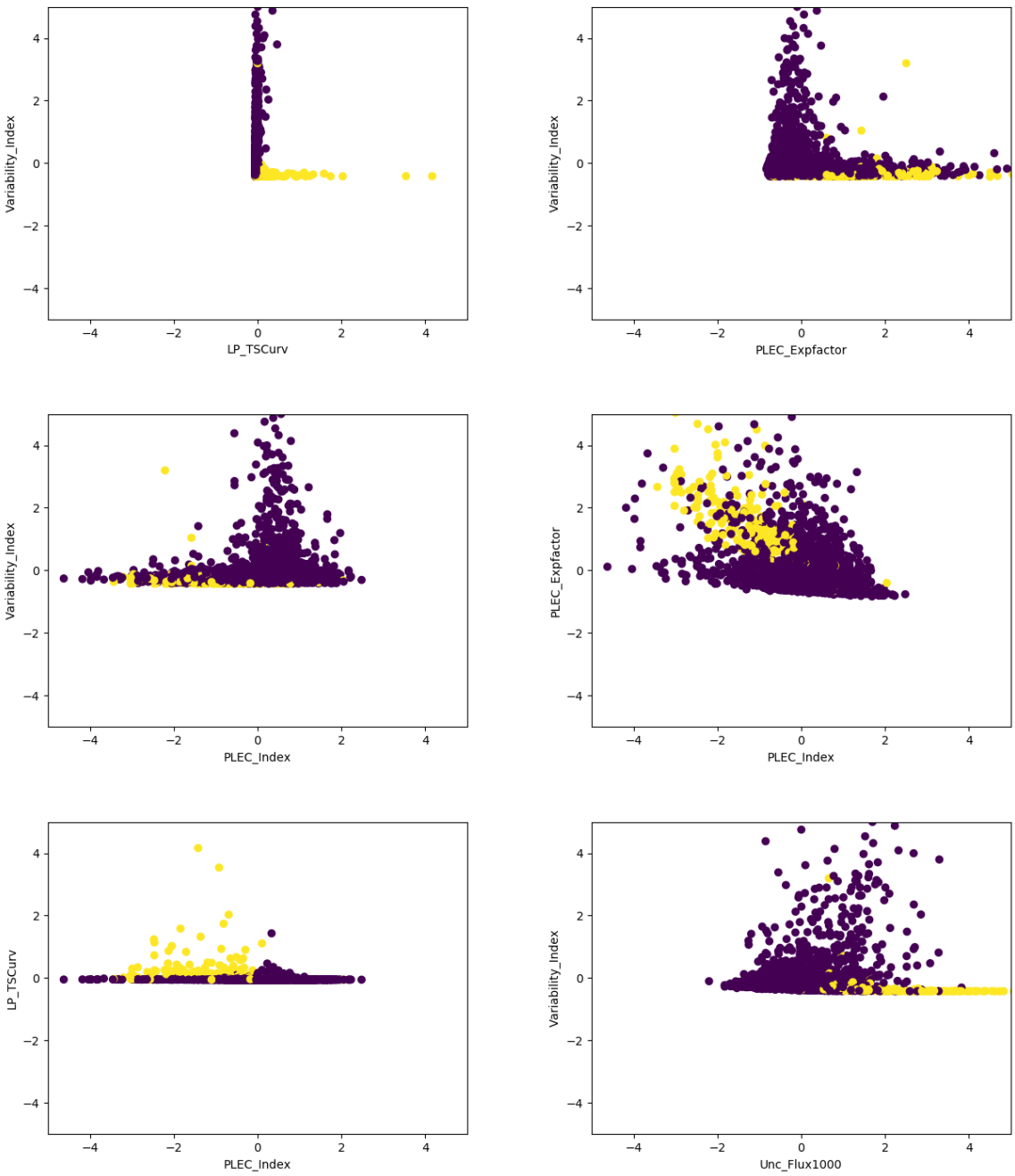

Figure 10: Yellow dot label pulsar and black dot label AGN

We only show the feature pairs with clear decision boundary. As shown in the figures, most of them can be separated by a linear or quartic function, and such decision boundary can be easily trained with our algorithms. Notice that we can only show at most two variable figure in the paper, with more feature, such high dimensional decision linear plane or non-linear

decision boundary can separate the data more accurately.

5.2 Physical interpretation

In this section, we will try to interpret the result shown above physically.

5.2.1 Variability

Active galactic nuclei have a giant black hole in the center region and an accretion disk around the black hole[9]. Such system will release a large amount of energy and make AGN one of the brightest objects in the universe. AGN luminosity can vary within few months or even days. This explains why we can classify AGN and pulsar with variability.
On the other hand,gamma-ray pulsar are steady sources. Therefore, gamma-ray pulsar has a much lower variability compared to AGN.

5.2.2 Spectral Shape

Besides the variability of the source, the spectral shape of the source is also a good predictor in our classification task.
The curvature of the spectral shapes is one of the key predictors in our classification task. LP and PLEC mean two different way to model the spectrum.
For LP:

$$\frac{dN}{dE} = K(\frac{E}{E_0})^{-\alpha-\beta log\frac{E}{E_0}}$$

For PLEC:

$$\frac{dN}{dE} = K(\frac{E}{E_0})^{-\Gamma} exp(a(E_0^b - E_0^a)$$

and LP_TSCurv and PLEC_TSCurv is the test statistic of curvature of the spectrum, the formula is

$$2log(\mathscr{L}(curvespectrum)/\mathscr{L}(powerlaw))$$

As we can see in fig.10, pulsar has a much higher test statistic in the curvature of the spectrum. The act a good predictor for our classification task.

6 Future prospects

The resulting catalog can be used to search for new pulsars from gamma-ray or different wavelengths like X-ray and radio. We have given a list of candidates to five-hundred-meter aperture spherical radio telescope(FAST) collaboration and they are performing searches

Figure 11: Five-hundred-meter aperture spherical radio telescope(FAST)

for the radio pulsars coincident with gamma-ray source which our algorithm predicts as a pulsar.

Five-hundred-meter aperture spherical radio telescope(FAST) is the largest single-dish radio telescope and it has unprecedented sensitivity. The FAST collaboration had recently joined the Fermi Pulsar Search Consortium (PSC).The Fermi Pulsar Search Consortium (PSC) is an international collaboration of astronomers working with radio telescope and members of the Fermi Large Area Telescope (LAT) collaboration. The goal of the PSC is to search for new gamma-ray pulsars from Fermi LAT point source.We rank the gamma-ray sources by significance, select those classified as Pulsar by our models and within declination limit of FAST and created a list of candidates.

One of the candidates FL8YJ0318.2+0254(3FGLJ0318.1+0252) had already resulted in the discovery of the radio pulsation by FAST team and then also gamma-ray pulsation. The source is an isolated millisecond pulsar(MSP), the first MSP discovered by FAST. After the installation of 19-beam feed-horn array, we expect FAST can help discover more gamma-ray pulsars.

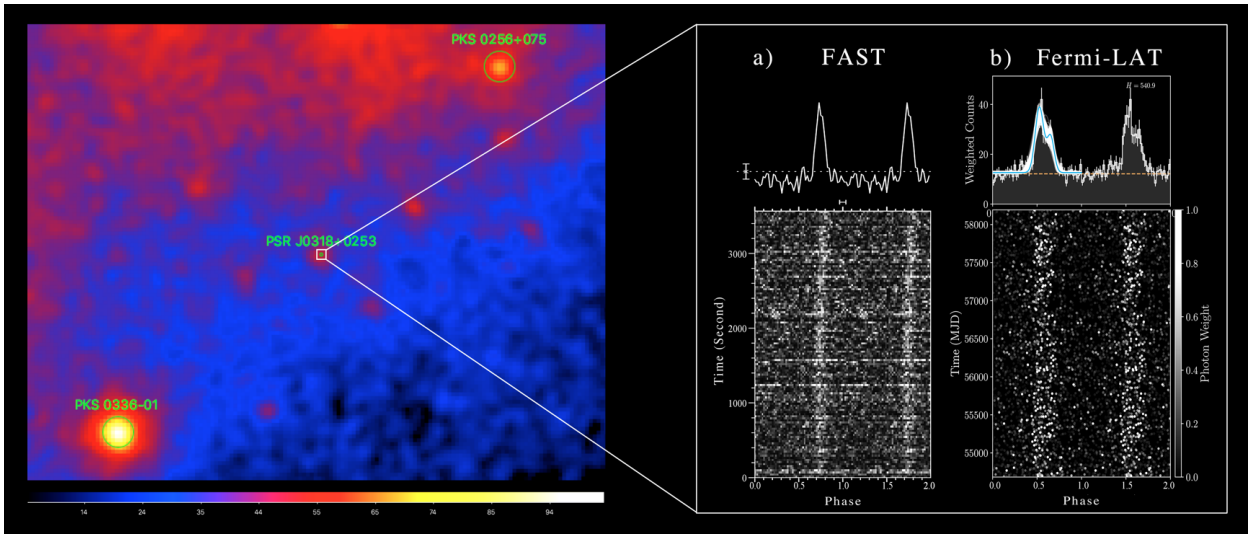

Figure 12: Gamma-ray image of FL8YJ0318.2+0254(3FGLJ0318.1+0252) (Fermi LAT collaboration and FAST collaboration)

7 Summary and discussion

From 20 MeV to 300 GeV,Fermi LAT is by far the most sensitive gamma-ray telescope. Fermi has helped astrophysicists solve a lot of longstanding problems in astronomy and discovered a lot of new astrophysical phenomena. The source catalog of Fermi LAT can help astrophysicists carry out a detailed analysis of the celestial objects. In Fermi FY8L catalog, over 2000 sources are unassociated. The classification of the nature of these unassociated sources is an important task for astrophysicists. We applied four different machine learning algorithms to this classification problem using the information given in the catalogs. The accuracy is over 95.8% for all four algorithms. We further examined the physical meaning of the features by visualization of the data and found the variability of the source and shape of the spectrum play a very important role in the classification.

Application of machine learning in astronomy and physics is a very fast growing interdisciplinary research field. In the age of big science, the detectors create a tremendous amount of data every day. By combining the techniques in statistics and data science, we can solve problems in physics and astronomy which are otherwise much harder to solve before.

Table 4: List of targets sent to FAST based on declination constraints and ranked by significance

	Source Name	RAJ2000	DEJ2000	GLON	GLAT	95%SM Axis	Name 3FGL	Signif Avg
1	FL8YJ1625.1-0020	246.28	-0.35	13.89	31.84	0.02	3FGLJ1625.1-0021	54.54
2	FL8YJ1225.9+2951	186.49	29.86	185.42	83.77	0.03	3FGLJ1225.9+2953	33.35
3	FL8YJ1120.6+0713	170.16	7.22	251.51	60.68	0.04	3FGLJ1120.6+0713	23.17
4	FL8YJ0318.2+0254	49.57	2.91	178.43	-43.59	0.04	3FGLJ0318.1+0252	21.76
5	FL8YJ1627.7+3219	246.95	32.32	52.99	43.23	0.04	3FGLJ1627.8+3217	18.05
6	FL8YJ1535.9+3744	233.99	37.74	60.63	54.02	0.04		17.55
7	FL8YJ1827.5+1141	276.89	11.69	40.76	10.54	0.04	3FGLJ1827.7+1141	15.45
8	FL8YJ2027.1+2811	306.78	28.19	68.84	-5.89	0.05	3FGLJ2026.8+2813	14.94
9	FL8YJ1747.5+0324	266.89	3.4	28.62	15.75	0.05	3FGLJ1747.3+0324	14.52
10	FL8YJ1842.1+2737	280.54	27.62	57.05	14.08	0.05	3FGLJ1842.2+2742	14.25
11	FL8YJ1612.0+1407	243.01	14.12	27.85	41.62	0.06	3FGLJ1611.9+1404	14.22
12	FL8YJ1602.2+2306	240.56	23.1	38.55	46.87	0.05	3FGLJ1601.9+2306	13.74
13	FL8YJ2212.4+0707	333.12	7.13	68.78	-38.5	0.05	3FGLJ2212.5+0703	13.71
14	FL8YJ2250.5+3307	342.64	33.12	95.62	-23.26	0.07	3FGLJ2250.6+3308	13.63
15	FL8YJ1818.5+1316	274.65	13.27	41.26	13.2	0.05	3FGLJ1818.5+1320	12.49
16	FL8YJ1828.6+3231	277.17	32.52	60.62	18.64	0.06	3FGLJ1829.2+3229	12.31
17	FL8YJ2116.2+3702	319.07	37.04	82.32	-8.34	0.08		11.74
18	FL8YJ2006.5+0147	301.63	1.8	43.38	-15.78	0.05	3FGLJ2006.6+0150	11.21
19	FL8YJ1059.4+0226	164.86	2.45	250.77	53.57	0.06	3FGLJ1059.3+0224	10.28
20	FL8YJ0652.9+4708	103.24	47.13	169.22	19.77	0.09		10.27
21	FL8YJ0003.6+3059	0.9	30.98	111	-30.78	0.09	3FGLJ0003.4+3100	10.13
22	FL8YJ1950.8+1211	297.71	12.2	50.67	-7.27	0.07	3FGLJ1950.2+1215	10.07
23	FL8YJ1814.4+3132	273.61	31.54	58.53	21.13	0.05		9.73
24	FL8YJ1527.8+1013	231.95	10.23	15.89	49.53	0.1		9.72
25	FL8YJ1304.4+1203	196.11	12.06	315.01	74.62	0.08	3FGLJ1304.6+1200	9.65
26	FL8YJ0546.6+0026	86.67	0.44	204.99	-14.16	0.34	3FGLJ0546.4+0031c	9.41
27	FL8YJ0506.0+5028	76.52	50.48	157.9	5.76	0.06		8.73
28	FL8YJ0427.8+3631	66.96	36.52	164.1	-8.51	0.12		7.81
29	FL8YJ1402.6+1306	210.67	13.1	356.36	68.22	0.11	3FGLJ1403.1+1304	7.78
30	FL8YJ1834.1+0615	278.54	6.26	36.57	6.68	0.13		7.73
31	FL8YJ1634.8+0235	248.7	2.6	18.33	31.32	0.09		7.51
32	FL8YJ1733.3+2235	263.34	22.59	45.96	26.72	0.1		7.29
33	FL8YJ1716.3+3434	259.08	34.57	58.07	33.66	0.1		6.8
34	FL8YJ0430.6+3533	67.67	35.55	165.21	-8.75	0.12		6.74
35	FL8YJ0342.3+3154	55.58	31.91	160.28	-18.3	0.14	3FGLJ0342.3+3148c	6.32
36	FL8YJ1823.2+1208	275.82	12.15	40.71	11.69	0.1		6.13
37	FL8YJ0431.3+3500	67.85	35.01	165.71	-9.01	0.2	3FGLJ0431.7+3503	6.11
38	FL8YJ0344.4+3202	56.1	32.04	160.55	-17.92	0.15		5.87
39	FL8YJ2038.7+3211	309.69	32.19	73.58	-5.57	0.17		5.87
40	FL8YJ1102.4+0245	165.6	2.75	251.3	54.3	0.15		5.25
41	FL8YJ2241.2+4301	340.32	43.03	99.05	-13.72	0.14		4.59

References

[1] BP Abbott et al. "Multi-messenger observations of a binary neutron star merger". In: *Astrophysical Journal Letters* 848.2 (2017), p. L12.

[2] Aous A Abdo et al. "Fermi large area telescope first source catalog". In: *The Astrophysical Journal Supplement Series* 188.2 (2010), p. 405.

[3] Aous A Abdo et al. "Fermi/large area telescope bright gamma-ray source list". In: *The Astrophysical Journal Supplement Series* 183.1 (2009), p. 46.

[4] Fabio Acero et al. "Fermi large area telescope third source catalog". In: *The Astrophysical Journal Supplement Series* 218.2 (2015), p. 23.

[5] WB Atwood et al. "The large area telescope on the Fermi gamma-ray space telescope mission". In: *The Astrophysical Journal* 697.2 (2009), p. 1071.

[6] Luca Baldini. "Space-Based Cosmic-Ray and Gamma-Ray Detectors: a Review". In: *arXiv preprint arXiv:1407.7631* (2014).

[7] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[8] Patrizia A Caraveo. "Gamma ray pulsar revolution". In: *Annual Review of Astronomy and Astrophysics* 52 (2014), pp. 211–250.

[9] Bradley W Carroll and Dale A Ostlie. *An introduction to modern astrophysics*. Cambridge University Press, 2017.

[10] David R Cox. "The regression analysis of binary sequences". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1958), pp. 215–242.

[11] Bernard Degrange and Gérard Fontaine. "Introduction to high-energy gamma-ray astronomy". In: *Comptes Rendus Physique* 16.6 (2015), pp. 587–599.

[12] Guillaume Dubus. "Gamma-ray binaries and related systems". In: *The Astronomy and Astrophysics Review* 21.1 (2013), p. 64.

[13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York, 2001.

[14] RC Hartman et al. "The third EGRET catalog of high-energy gamma-ray sources". In: *The Astrophysical Journal Supplement Series* 123.1 (1999), p. 79.

[15] Henric Krawczynski and Ezequiel Treister. "Active galactic nuclei the physics of individual sources and the cosmic history of formation and evolution". In: *Frontiers of Physics* 8.6 (2013), pp. 609–629.

[16] Lupin Chun-Che Lin. "Radio-quiet Gamma-ray Pulsars". In: *Journal of Astronomy and Space Sciences* 33.3 (2016), pp. 147–166.

[17] PL Nolan et al. “Fermi large area telescope second source catalog”. In: *The Astrophysical Journal Supplement Series* 199.2 (2012), p. 31.

[18] PM Saz Parkinson et al. “Classification and ranking of Fermi LAT gamma-ray sources from the 3FGL catalog using machine learning techniques”. In: *The Astrophysical Journal* 820.1 (2016), p. 8.

[19] PM Saz Parkinson et al. “Eight γ-ray pulsars discovered in blind frequency searches of Fermi LAT data”. In: *The Astrophysical Journal* 725.1 (2010), p. 571.

Appendices

The code of variability index generation

```
# -*- coding: utf-8 -*-
"""
Created on Fri Oct  6 15:41:26 2017

@author: kwoklung
"""
#The light curve file should end with .out and there should be a FL8Yorg.csv file which have all the column but variability index missing.The execution result in a FL8Yorgnew.csv file.
import os
import scipy.stats
import numpy as np
import glob
import pandas as pd
from multiprocessing import Pool

name="*.out"
lis=np.array(glob.glob(name))
result=np.empty((len(lis),2),dtype="<U25")
#def test(i):
for i in range(len(lis)):
    result[i,0]=(lis[i].replace("lc_","")).replace(".out","")
    temp=np.loadtxt(lis[i],delimiter=' ')
    expect=np.sum(temp[:,1])/len(temp)
    de=0
    #std=np.std(temp[:,1])
    count=0
    for j in range(96):
    #result[i,1]=de
        su=((temp[j,1]-expect)**2)/(expect**2)
        if su <3:
            de+=su
            count+=1
    result[i,1]=de*96.0/count
    if i %50==0:
        print (i)


for i in range(len(result)):
    result[i,0]=result[i,0].replace("p",'+')
np.savetxt("var.csv",result,delimiter=',',fmt="%18s")
train=np.loadtxt("FL8Yorg.csv",skiprows=1,delimiter=',',dtype="<U18")


```

```
for i in range(len(train)):
    train[i,0]=train[i,0].replace(" ",'')
    train[i][29]=result[np.argwhere(result[:,0]==train[i,0])][0][0][1]

tr=pd.DataFrame(train)
tr.to_csv("FL8Yneworg.csv",header=["Source_Name","RAJ2000","DEJ2000","GLON","GLAT","Signif_Avg"
    ,"Pivot_Energy","Flux_Density","Unc_Flux_Density","Flux100","Unc_Flux100","Flux1000","
    Unc_Flux1000","Energy_Flux100","Unc_Energy_Flux100","Energy_Flux1000","Unc_Energy_Flux1000"
    ,"PL_Index","Unc_PL_Index","PLEC_Index","Unc_PLEC_Index","LP_INDEX","Unc_LP_Index","
    LP_TSCurv","PLEC_TSCurv","PLEC_Expfactor","Unc_PLEC_Expfactor","LP_beta","Unc_LP_beta","
    Variability_Index","CLASS","CLASS"],index=False)
```

The code for evaluate the performance

```
'''
This file will run RF first and output the accuracy
Then scale the dataset to unit variance
Then LR
Then ANN
Then support vector machine
All of them use 10-fold cross validation except ANN
All of them use class weight
The dataset is call FL8Ynewtrain.csv by default
Packages required: numpy keras tensorflow pandas
'''
# -*- coding: utf-8 -*-
from math import sqrt
import numpy as np
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn import linear_model
from sklearn.metrics import roc_curve, auc
import sklearn
import sys
if len(sys.argv) >= 2:
  seed = int(sys.argv[1])
else:
  seed = 0
np.random.seed(seed)
from keras.models import Model, Sequential
from keras.layers import Dense, GlobalAveragePooling2D,Dropout, Flatten,Input
from keras import backend as K
from keras.metrics import top_k_categorical_accuracy, sparse_top_k_categorical_accuracy
from keras import applications
from keras import regularizers
from sklearn import preprocessing
import keras
from keras.objectives import categorical_crossentropy
```

```
from sklearn.model_selection import cross_val_score
import pandas as pd
import math
import matplotlib.pyplot as plt
from sklearn.ensemble import BaggingClassifier
from sklearn import preprocessing
from sklearn.svm import SVC
DIR = "./FL8Ynewtrain.csv"
COLUMNS = ["CLASS","Signif_Avg","Npred","Pivot_Energy","Flux_Density","Unc_Flux_Density","
    Flux100","Unc_Flux100","Flux1000","Unc_Flux1000","Energy_Flux100","Unc_Energy_Flux100","
    Energy_Flux1000","Unc_Energy_Flux1000","PL_Index","Unc_PL_Index","PLEC_Index","LP_Index","
    Unc_LP_Index","LP_TSCurv","PLEC_TSCurv","PLEC_Expfactor","Unc_PLEC_Expfactor","LP_beta","
    Variability_Index"]
FEATURES = ["Signif_Avg","Npred","Pivot_Energy","Flux_Density","Unc_Flux_Density","Flux100","
    Unc_Flux100","Flux1000","Unc_Flux1000","Energy_Flux100","Unc_Energy_Flux100","
    Energy_Flux1000","Unc_Energy_Flux1000","PL_Index","Unc_PL_Index","PLEC_Index","LP_Index","
    Unc_LP_Index","LP_TSCurv","PLEC_TSCurv","PLEC_Expfactor","Unc_PLEC_Expfactor","LP_beta","
    Variability_Index"]
LABEL = "CLASS"
def accuracy():
      count = 0
      correct = 0
      for i in range(0,predicted_class.shape[0]):
          count += 1
          if predicted_class[i] == prediction_set[i, 0]:
              correct += 1
      acc = correct/count
      print("Accuracy: " + str(acc))
      print("Total matches: " + str(correct))
      count = 0
      correct = 0
      for i in range(0,predicted_class.shape[0]):
          if prediction_set[i, 0]==1:
              count += 1
          if predicted_class[i] == prediction_set[i, 0] and prediction_set[i, 0]==1:
              correct += 1
      acc = correct/count
      print("PSR Accuracy: " + str(acc))
      print("Total matches: " + str(correct))
      count = 0
      correct = 0
      for i in range(0,predicted_class.shape[0]):
          if prediction_set[i, 0]==0:
              count += 1
          if predicted_class[i] == prediction_set[i, 0] and prediction_set[i, 0]==0:
              correct += 1
      acc = correct/count
```

```
74        print("AGN Accuracy: " + str(acc))
75        print("Total matches: " + str(correct))
76
77  def newaccuracy():
78      psrcorrect=0
79      psrwrong=0
80      agncorrect=0
81      agnwrong=0
82      for i in range(0,predicted_class.shape[0]):
83          if predicted_class[i]==0 and prediction_set[i, 0]==0:
84              agncorrect+=1
85          if predicted_class[i]==0 and prediction_set[i, 0]==1:
86              agnwrong +=1
87          if predicted_class[i]==1 and prediction_set[i, 0]==1:
88              psrcorrect+=1
89          if predicted_class[i]==1 and prediction_set[i, 0]==0:
90              psrwrong +=1
91      print("agncorrect:"+str(agncorrect))
92      print("agnwrong:"+str(agnwrong))
93      print("psrcorrect:"+str(psrcorrect))
94      print("psrwrong:"+str(psrwrong))
95
96  def predictclass(x,threshold=0.5):
97      preclass=np.zeros((len(x),1))
98      for i in range(len(x)):
99          if x[i]>threshold:
100             preclass[i]=1
101         else:
102             preclass[i]=0
103     return preclass
104
105 temp=np.array([0,3,4,5,6,8,12,13,14, 15,17,18,19,20,22,24]) #select the subset of feature
106 all_set = pd.read_csv(DIR, skipinitialspace=True,
107                           skiprows=0, usecols=COLUMNS).as_matrix()
108 all_set=all_set[:,temp]
109 np.random.shuffle(all_set) #shuffle the whole dataset
110 training_set=all_set[0:math.floor(all_set.shape[0]*0.7)] #Split the dataset
111 prediction_set=all_set[math.floor(all_set.shape[0]*0.7):]
112
113 kf = KFold(n_splits=10) # doing 10-fold
114 kf.get_n_splits(training_set)
115 result=np.zeros((training_set.shape[0],3))
116 for train_index, test_index in kf.split(training_set):
117     clf=RandomForestClassifier(n_estimators=500,criterion='gini',n_jobs=-1,bootstrap=True,
        class_weight="balanced_subsample")
118     clf.fit(training_set[train_index][:,1:],training_set[train_index][:,0])
119     result[test_index]=np.vstack((clf.predict(training_set[test_index][:,1:]),clf.predict_proba
```

```
    (training_set[test_index][:,1:])[:,0],clf.predict_proba(training_set[test_index][:,1:])
    [:,1])).T

false_positive_rate, true_positive_rate, thresholds = roc_curve(training_set[:,0], result[:,2])
roc_auc = auc(false_positive_rate, true_positive_rate)
best=thresholds[np.argmin(false_positive_rate-true_positive_rate)] #obtain the best thresholds

clf=RandomForestClassifier(n_estimators=500,criterion='gini',n_jobs=-1,bootstrap=True,
    class_weight="balanced_subsample")


clf.fit(training_set[:,1:],training_set[:,0])
predicted_prob=clf.predict_proba(prediction_set[:,1:])
predicted_class=np.zeros(prediction_set.shape[0])
for i in range(prediction_set.shape[0]):
    if predicted_prob[i][1]>best:
        predicted_class[i]=1
print("result of random forest with boostrap : ")
print("best threshold : "+str(best))
rfwithboostrap=predicted_class.reshape((len(predicted_class),1))
newaccuracy()
accuracy()



all_set[:,1:]=preprocessing.scale(all_set[:,1:]) #scale the dataset and split it again
training_set=all_set[0:math.floor(all_set.shape[0]*0.7)] #Notice that the training set and test
     set is still the same except the scaled features since we didn't shuffle it.
prediction_set=all_set[math.floor(all_set.shape[0]*0.7):]
kf = KFold(n_splits=10)
kf.get_n_splits(training_set)
result=np.zeros((training_set.shape[0],3))
for train_index, test_index in kf.split(training_set):
    clf=linear_model.LogisticRegression(class_weight="balanced")
    clf.fit(training_set[train_index][:,1:],training_set[train_index][:,0])
    result[test_index]=np.vstack((clf.predict(training_set[test_index][:,1:]),clf.predict_proba
    (training_set[test_index][:,1:])[:,0],clf.predict_proba(training_set[test_index][:,1:])
    [:,1])).T

false_positive_rate, true_positive_rate, thresholds = roc_curve(training_set[:,0], result[:,2])
roc_auc = auc(false_positive_rate, true_positive_rate)
best=thresholds[np.argmin(false_positive_rate-true_positive_rate)]
class_weight = {0 : 1.,1: len(training_set)/np.count_nonzero(training_set[:,0]==1),}
clf=linear_model.LogisticRegression(class_weight="balanced")
clf.fit(training_set[:,1:],training_set[:,0])
predicted_prob=clf.predict_proba(prediction_set[:,1:])
predicted_class=np.zeros(prediction_set.shape[0])
```

```
for i in range(prediction_set.shape[0]):
    if predicted_prob[i][1]>best:
        predicted_class[i]=1

print("result of logistic regression : ")
print("best threshold : "+str(best))
logistic=predicted_class.reshape((len(predicted_class),1))
newaccuracy()
accuracy()

#For neural network, we didn't apply 10-fold since the training step and sample size itself affect the output probability .
model=Sequential() #Define the model
best=0.5
adam=keras.optimizers.Adam(lr=0.002)
model.add(Dense(15,input_shape=(training_set.shape[1]-1,),activation='sigmoid', kernel_regularizer=regularizers.l2(0.003))) #Add kernel_regularizer
model.add(Dense(9,activation='sigmoid',kernel_regularizer=regularizers.l2(0.003)))
model.add(Dense(1, activation='sigmoid'))
class_weight = {0 : 1.,1: len(training_set)/np.count_nonzero(training_set[:,0]==1),}  # calculate the class_weight
model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['binary_accuracy'])
model.fit(training_set[:,1:],training_set[:,0],epochs=4000,batch_size=len(training_set), class_weight=class_weight,verbose=0,shuffle=False) #turn off the shuffle for reproducible result
predicted_prob=model.predict_proba(prediction_set[:,1:])
predicted_class=predictclass(predicted_prob,best)
print("result of ANN : ")
print("best threshold : "+str(best))
newaccuracy()
accuracy()
ANN=predicted_class.reshape((len(predicted_class),1))

###To get the best thresholds
kf = KFold(n_splits=10)
kf.get_n_splits(training_set)
result=np.zeros((training_set.shape[0],3))
for train_index, test_index in kf.split(training_set):
    class_weight = {0 : 1.,1: len(training_set[train_index])/np.count_nonzero(training_set[train_index][:,0]==1),}
    clf=SVC(C=5,kernel="linear",max_iter=30000,probability=True,class_weight=class_weight)
    clf.fit(training_set[train_index][:,1:],training_set[train_index][:,0])
    result[test_index]=np.vstack((clf.predict(training_set[test_index][:,1:]),clf.predict_proba(training_set[test_index][:,1:])[:,0],clf.predict_proba(training_set[test_index][:,1:])[:,1])).T

false_positive_rate, true_positive_rate, thresholds = roc_curve(training_set[:,0], result[:,2])
```

```
roc_auc = auc(false_positive_rate, true_positive_rate)
best=thresholds[np.argmin(false_positive_rate-true_positive_rate)]
class_weight = {0 : 1.,1: len(training_set)/np.count_nonzero(training_set[:,0]==1),} #calculate the class_weight
training_set[:,1:]=preprocessing.scale(training_set[:,1:])
prediction_set[:,1:]=preprocessing.scale(prediction_set[:,1:])
clf=SVC(C=5,kernel="linear",max_iter=30000,probability=True,class_weight=class_weight)
clf.fit(training_set[:,1:],training_set[:,0])
predicted_prob=clf.predict_proba(prediction_set[:,1:])
predicted_class=np.zeros(prediction_set.shape[0])
for i in range(prediction_set.shape[0]):
    if predicted_prob[i][1]>best:
        predicted_class[i]=1

print("result of SVM : ")
print("best threshold : "+str(best))
newaccuracy()
accuracy()
svm=predicted_class.reshape((len(predicted_class),1))



psrdidnotmatch=0
agndidnotmatch=0
for i in range(len(logistic)):
    if (logistic[i]==0 or rfwithboostrap[i]==0 or svm[i]==0 or ANN[i]==0) and prediction_set[i,0]==1:
        psrdidnotmatch+=1
    if (logistic[i]==1 or rfwithboostrap[i]==1 or svm[i]==1 or ANN[i]==1) and prediction_set[i,0]==1:
        agndidnotmatch+=1

print(psrdidnotmatch)
print(agndidnotmatch)
psrmatch=0
agnmatch=0
for i in range(len(logistic)):
    if logistic[i]==rfwithboostrap[i] and svm[i]==rfwithboostrap[i] and ANN[i]==svm[i] and logistic[i]==1:
        psrmatch+=1
    if logistic[i]==rfwithboostrap[i] and svm[i]==rfwithboostrap[i] and ANN[i]==svm[i] and logistic[i]==0:
        agnmatch+=1

print(psrmatch)
print(agnmatch)
```

The code for predicting the result

```
'''
This file will run RF first and output the accuracy
Then scale the dataset to unit variance
Then LR
Then ANN
Then support vector machine
All of them use 10-fold cross validation except ANN
All of them use class weight
The dataset is call FL8Ynewtrain.csv by default
Packages required: numpy keras tensorflow pandas
'''

# -*- coding: utf-8 -*-
"""
Created on Wed Jul 19 14:14:37 2017

@author: user98
"""
from math import sqrt
import numpy as np
np.random.seed(0)
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn import linear_model
from sklearn.metrics import roc_curve, auc
import sklearn
from keras.models import Model, Sequential
from keras.layers import Dense, GlobalAveragePooling2D,Dropout, Flatten,Input
from keras import backend as K
from keras.metrics import top_k_categorical_accuracy, sparse_top_k_categorical_accuracy
from keras import applications
from keras import regularizers
from sklearn import preprocessing
import keras
from keras.objectives import categorical_crossentropy
from sklearn.model_selection import cross_val_score
import pandas as pd
import math
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import BaggingClassifier
from sklearn import preprocessing
from sklearn.svm import SVC
TRAINDIR="./FL8Ynewtrain.csv"
TESTDIR="./FL8Yfull.csv"
def predictclass(x,threshold=0.5):
    preclass=np.zeros((len(x),1))
```

```
    for i in range(len(x)):
        if x[i]>threshold:
            preclass[i]=1
        else:
            preclass[i]=0
    return preclass

COLUMNS = ["CLASS","Signif_Avg","Npred","Pivot_Energy","Flux_Density","Unc_Flux_Density","
    Flux100","Unc_Flux100","Flux1000","Unc_Flux1000","Energy_Flux100","Unc_Energy_Flux100","
    Energy_Flux1000","Unc_Energy_Flux1000","PL_Index","Unc_PL_Index","PLEC_Index","LP_Index","
    Unc_LP_Index","LP_TSCurv","PLEC_TSCurv","PLEC_Expfactor","Unc_PLEC_Expfactor","LP_beta","
    Variability_Index"]
FEATURES = ["Signif_Avg","Npred","Pivot_Energy","Flux_Density","Unc_Flux_Density","Flux100","
    Unc_Flux100","Flux1000","Unc_Flux1000","Energy_Flux100","Unc_Energy_Flux100","
    Energy_Flux1000","Unc_Energy_Flux1000","PL_Index","Unc_PL_Index","PLEC_Index","LP_Index","
    Unc_LP_Index","LP_TSCurv","PLEC_TSCurv","PLEC_Expfactor","Unc_PLEC_Expfactor","LP_beta","
    Variability_Index"]
LABEL = "CLASS1"

re=pd.read_csv(TESTDIR, skipinitialspace=True,skiprows=0).as_matrix()
temp=np.array([0, 1,2,3,4,5,6,7,8,9,10,11,12,13,14, 15,16,17,18,19,20,21,22,24])
training_set=pd.read_csv(TRAINDIR, skipinitialspace=True,
                         skiprows=0, usecols=COLUMNS).as_matrix()
np.random.shuffle(training_set)

prediction_set= pd.read_csv(TESTDIR, skipinitialspace=True,
                         skiprows=0, usecols=COLUMNS).as_matrix()

training_set=training_set[:,temp] #remove the unwant feature(Here only LP_beta is removed)
prediction_set=prediction_set[:,temp]



kf = KFold(n_splits=10) #10-fold
kf.get_n_splits(training_set)
result=np.zeros((training_set.shape[0],3))
for train_index, test_index in kf.split(training_set):
    clf=RandomForestClassifier(n_estimators=500,criterion='gini',n_jobs=-1,bootstrap=True,
    class_weight="balanced_subsample")
    clf.fit(training_set[train_index][:,1:],training_set[train_index][:,0])
    result[test_index]=np.vstack((clf.predict(training_set[test_index][:,1:]),clf.predict_proba
    (training_set[test_index][:,1:])[:,0],clf.predict_proba(training_set[test_index][:,1:])
    [:,1])).T

false_positive_rate, true_positive_rate, thresholds = roc_curve(training_set[:,0], result[:,2])
roc_auc = auc(false_positive_rate, true_positive_rate)
best=thresholds[np.argmin(false_positive_rate-true_positive_rate)] #get the best threshold
```

```
84
85 clf=RandomForestClassifier(n_estimators=500,criterion='gini',n_jobs=-1,bootstrap=True,
       class_weight="balanced_subsample")
86
87
88 clf.fit(training_set[:,1:],training_set[:,0])
89 predicted_prob=clf.predict_proba(prediction_set[:,1:])
90 predicted_class=np.zeros(prediction_set.shape[0])
91 for i in range(prediction_set.shape[0]):
92     if predicted_prob[i][1]>best:
93         predicted_class[i]=1
94 print("result of random forest with boostrap : ")
95 print("best threshold : "+str(best))
96 rfwithboostrap=predicted_class.reshape((len(re),1))   #This is the result
97
98 #The part is to scale the whole dataset to unit variance and zero mean
99 all_set=np.vstack((training_set[:,1:],prediction_set[:,1:]))
100 all_set=preprocessing.scale(all_set)
101 lens=len(training_set)
102 training_set[:,1:]=all_set[:lens,:]
103 prediction_set[:,1:]=all_set[lens:,:]
104
105
106 #logistic regression
107 kf = KFold(n_splits=10)
108 kf.get_n_splits(training_set)
109 result=np.zeros((training_set.shape[0],3))
110 for train_index, test_index in kf.split(training_set):
111     clf=linear_model.LogisticRegression(class_weight="balanced")
112     clf.fit(training_set[train_index][:,1:],training_set[train_index][:,0])
113     result[test_index]=np.vstack((clf.predict(training_set[test_index][:,1:]),clf.predict_proba
        (training_set[test_index][:,1:])[:,0],clf.predict_proba(training_set[test_index][:,1:])
        [:,1])).T
114
115 false_positive_rate, true_positive_rate, thresholds = roc_curve(training_set[:,0], result[:,2])
116 roc_auc = auc(false_positive_rate, true_positive_rate)
117 best=thresholds[np.argmin(false_positive_rate-true_positive_rate)]
118 class_weight = {0 : 1.,1: len(training_set)/np.count_nonzero(training_set[:,0]==1),}
119 clf=linear_model.LogisticRegression(class_weight="balanced")
120 clf.fit(training_set[:,1:],training_set[:,0])
121 predicted_prob=clf.predict_proba(prediction_set[:,1:])
122 predicted_class=np.zeros(prediction_set.shape[0])
123 for i in range(prediction_set.shape[0]):
124     if predicted_prob[i][1]>best:
125         predicted_class[i]=1
126
127 print("result of logistic regression : ")
```

```
print("best threshold : "+str(best))
logistic=predicted_class.reshape((len(re),1))

best=0.5
model=Sequential()
adam=keras.optimizers.Adam(lr=0.002)
model.add(Dense(15,input_shape=(training_set.shape[1]-1,),activation='sigmoid',
    kernel_regularizer=regularizers.l2(0.003)))
model.add(Dense(9,activation='sigmoid',kernel_regularizer=regularizers.l2(0.003)))
model.add(Dense(6,activation='sigmoid',kernel_regularizer=regularizers.l2(0.003)))
model.add(Dense(1, activation='sigmoid'))
class_weight = {0 : 1.,1: len(training_set)/np.count_nonzero(training_set[:,0]==1),}
model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['binary_accuracy'])
model.fit(training_set[:,1:],training_set[:,0],epochs=4000,batch_size=len(training_set),
    class_weight=class_weight,verbose=0,shuffle=False)
predicted_prob=model.predict_proba(prediction_set[:,1:])
predicted_class=predictclass(predicted_prob,best)
print("result of ANN : ")
print("best threshold : "+str(best))
ANN=predicted_class.reshape((len(re),1))

kf = KFold(n_splits=10)
kf.get_n_splits(training_set)
result=np.zeros((training_set.shape[0],3))
for train_index, test_index in kf.split(training_set):
    class_weight = {0 : 1.,1: len(training_set[train_index])/np.count_nonzero(training_set[
    train_index][:,0]==1),}
    clf=SVC(C=5,kernel="linear",max_iter=30000,probability=True,class_weight=class_weight)
    clf.fit(training_set[train_index][:,1:],training_set[train_index][:,0])
    result[test_index]=np.vstack((clf.predict(training_set[test_index][:,1:]),clf.predict_proba
    (training_set[test_index][:,1:])[:,0],clf.predict_proba(training_set[test_index][:,1:])
    [:,1])).T

false_positive_rate, true_positive_rate, thresholds = roc_curve(training_set[:,0], result[:,2])
roc_auc = auc(false_positive_rate, true_positive_rate)
best=thresholds[np.argmin(false_positive_rate-true_positive_rate)]
class_weight = {0 : 1.,1: len(training_set)/np.count_nonzero(training_set[:,0]==1),}
clf=SVC(C=5,kernel="linear",max_iter=30000,probability=True,class_weight=class_weight)
clf.fit(training_set[:,1:],training_set[:,0])
predicted_prob=clf.predict_proba(prediction_set[:,1:])
predicted_class=np.zeros(prediction_set.shape[0])
for i in range(prediction_set.shape[0]):
    if predicted_prob[i][1]>best:
        predicted_class[i]=1
```

```
svm=predicted_class.reshape((len(re),1))

re=np.hstack((re,logistic,rfwithboostrap,ANN,svm))
re=pd.DataFrame(re)
re.to_csv("FL8Yresult.csv",header=["CLASS","Source_Name","RAJ2000","DEJ2000","GLON","GLAT","Conf_95_SemiMajor","Conf_95_SemiMinor","Conf_95_Posang","NickName","Name_3FGL","CLASS_3FGL","CLASS_4FGL",
"Signif_Avg","Npred","Pivot_Energy","Flux_Density","Unc_Flux_Density","Flux100","Unc_Flux100","Flux1000","Unc_Flux1000","Energy_Flux100","Unc_Energy_Flux100","Energy_Flux1000","Unc_Energy_Flux1000","PL_Index","Unc_PL_Index","PLEC_Index","LP_Index","Unc_LP_Index","LP_TSCurv","PLEC_TSCurv","PLEC_Expfactor","Unc_PLEC_Expfactor","LP_beta","Variability_Index","LR","RFwithboostrap","ANN","SVM"],index=False)
psrmatch=0
agnmatch=0
for i in range(len(re)):
    if logistic[i]==rfwithboostrap[i] and svm[i]==rfwithboostrap[i] and ANN[i]==svm[i] and logistic[i]==1:
        psrmatch+=1
    if logistic[i]==rfwithboostrap[i] and svm[i]==rfwithboostrap[i] and ANN[i]==svm[i] and logistic[i]==0:
        agnmatch+=1

print(psrmatch)
print(agnmatch)
```

The prediction of FL8Y

https://drive.google.com/file/d/1vwRWEIDNOqhv9iOt6dJebyH26XOoypx1/view?usp=sharing