Machine Learning Advanced Nanodegree

Capstone Proposal

Sayan Biswas

September 06, 2018

I. Definition

Project Overview

Pulsars are member of a family of stars, called neutron stars. They are highly dense spherical objects about the size of a large city (e.g. 20 km in diameter) with mass of several times that of Sun, which is of about 1.4 million km in diameter. They are called pulsars because they radiate two focused beams of light in opposite directions, which usually don't align with their axis of rotation (similar to a lighthouse). The frequencies of these pulses are so accurate that they can rival atomic clocks which are the most precise clocks known to us. The reason behind choosing this topic is to gain a good understanding on which features are observed in a pulsar, so that we can get better at finding them, and also given a bunch of relevant features associated to celestial objects, we can classify them as pulsars. In this project I did a comparative study of performance with multiple classifiers, which will help us to identify a pulsar, given the set of features very effectively. The Kaggle dataset can be found here.

Problem Statement

The objective here is to classify the celestial objects to two categories: pulsar and non-pulsar with very high accuracy and precision. This is important because pulsars serve many scientific purposes like below:

1. They give information about the physics of the matter inside them. Under such incredible pressure (second to only black holes) matter behaves quite differently. Hence, it works as very good resource materials, because such extreme environments can't be practically created (yet) in a test lab on earth.
2. Pulsars are extremely accurate in emitting their pulses, as mentioned before. Hence observing the changes in a pulsar's blinking provides information about what's happening in its vicinity. Also, these can be used as a clock for scientific experiments which require precise timing.
3. Pulsars are useful in finding extrasolar planets. In fact, the first extrasolar planet was found orbiting a pulsar.
4. They help in testing aspects of Albert Einstein's theory of general relativity.

Approach towards finding a solution

1. Choose a set of classifiers, spanning from simple to relatively complex ones.
2. Train these classifiers and see which one of these give the best performance.
3. While choosing among the classifiers, make sure we tune all hyperparameters, and only then compare among the optimal versions of each classifiers.
4. Use K-fold cross validation to minimize selection bias of test and train data. Run grid search on each fold for tuning and selecting best hyperparameters.

Metrics

Accuracy measures how often the classifier makes the correct prediction. It’s the ratio of the number of correct predictions to the total number of predictions. In other words it is the ratio of

[(True Positives + True Negatives)/(True Positives + True Negatives + False Positives + False Negatives)]

Precision tells us what proportion of our predictions are correct. It is a ratio of true positives (stars classified as pulsars, which are actually pulsars) to all positives (all stars classified as pulsars, irrespective of whether they are actually pulsar or not), in other words it is the ratio of

[True Positives/(True Positives + False Positives)]

Recall tells us what proportion of stars that actually were pulsars were classified by us as pulsars. It is a ratio of true positives (stars classified as pulsars, which are actually pulsars) to all the stars that were actually pulsars, in other words it is the ratio of

[True Positives/(True Positives + False Negatives)]

For classification problems that are skewed (like ours, which we'll see later) in their classification distributions, accuracy by itself is not a very good metric. For such cases, precision and recall become useful. These two metrics can be combined to get the F-beta score, which is a weighted average (harmonic mean with a factor beta) of the precision and recall scores. This score can range from 0 to 1, with 1 being the best possible F-beta score.

$$F_\beta=(1+\beta^2)\cdot precision\cdot recall/((\beta^2\cdot precision)+recall)$$

In our case, we'd consider accuracy and recall over precision as our performance metric, because in the dataset, percentage of pulsars are pretty low. So, we wouldn't want to lose out on detecting the actual pulsars, i.e. we better capture all the stars which are pulsars, it matters less if we classify some non-pulsars as pulsars. So, we'll choose the beta value as 2.

II. Analysis

Data Exploration

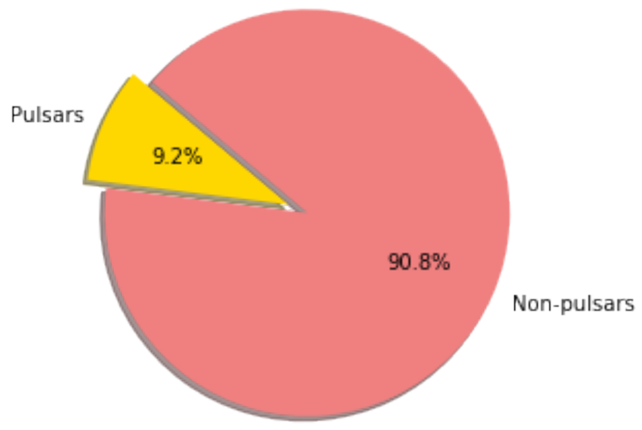
Our dataset consists of a csv file with almost 18000 entries, and it’s obtained from Kaggle dataset. The input feature set consists of 8 continuous variables mentioned below, whose names are also self-explanatory:

1. Mean of the integrated profile.
2. Standard deviation of the integrated profile.
3. Excess kurtosis of the integrated profile.
4. Skewness of the integrated profile.
5. Mean of the DM-SNR curve.
6. Standard deviation of the DM-SNR curve.

7. Excess kurtosis of the DM-SNR curve.
8. Skewness of the DM-SNR curve.

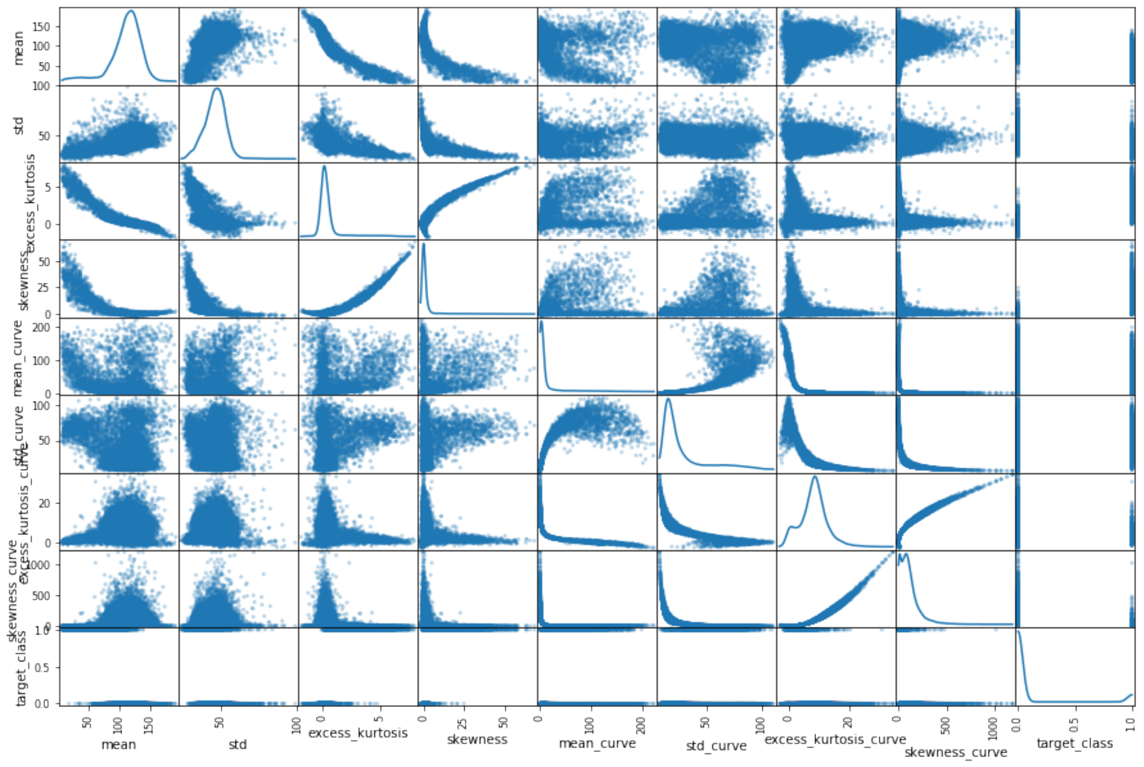
This is a binary classification problem. So based on these features we need to decide whether the object is a pulsar or not. This metric is given by the variable class, where 1 means pulsar, and 0 otherwise. Without these metrices, detecting real pulsars from false ones would become tricky, because many of the pulses detected through instruments are from noise and radio frequency interference.

Exploratory Visualization

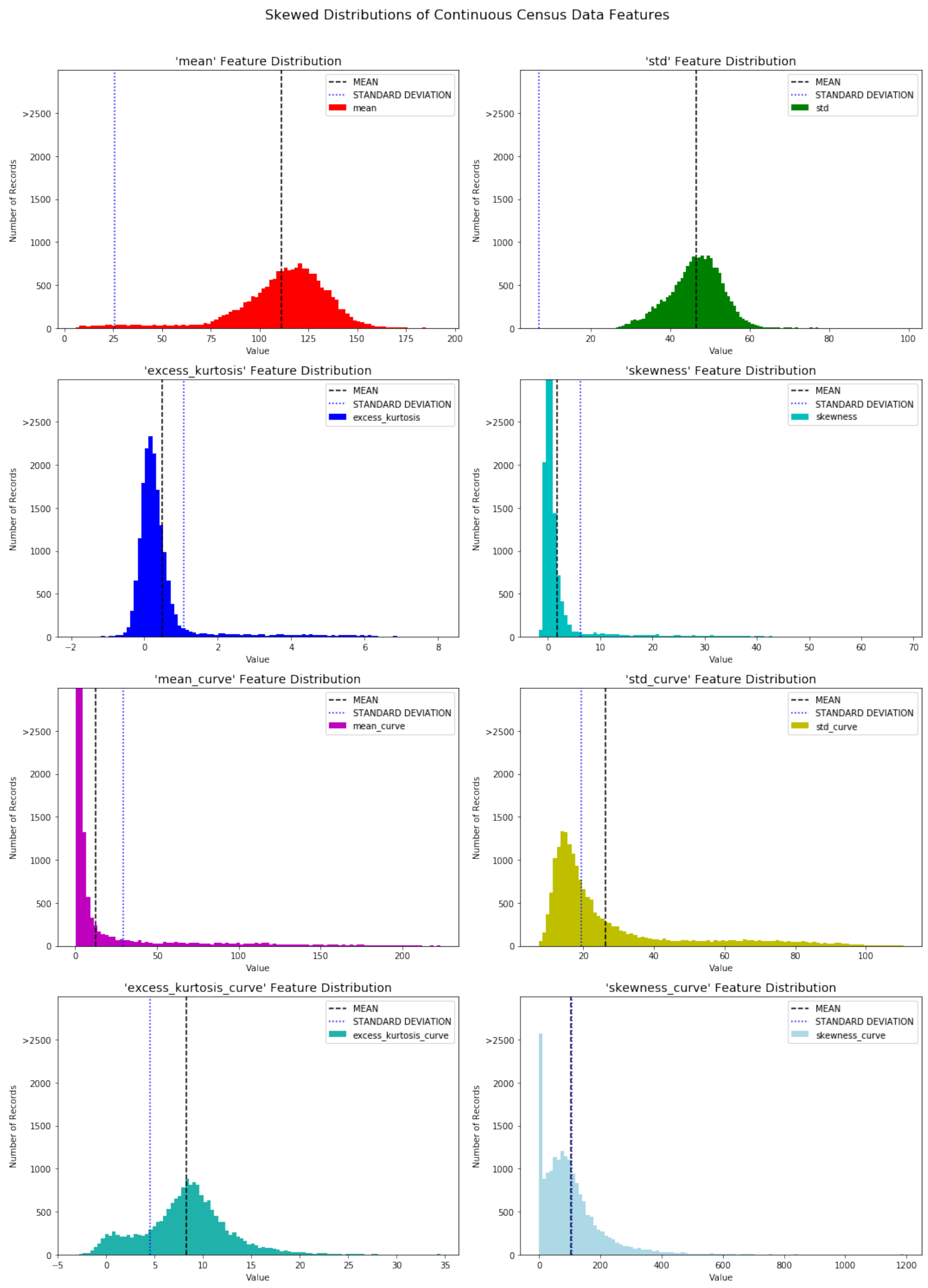


```
Total number of records: 17898
Total number of pulsars: 1639
Total number of non-pulsars: 16259
Percentage of pulsars in the dataset: 9.15744775953%
```

We can see only around 9% of the data are pulsars, rest 91% are non-pulsars. Let's look at some visualizations which will help us get an idea of the distribution of all input features.

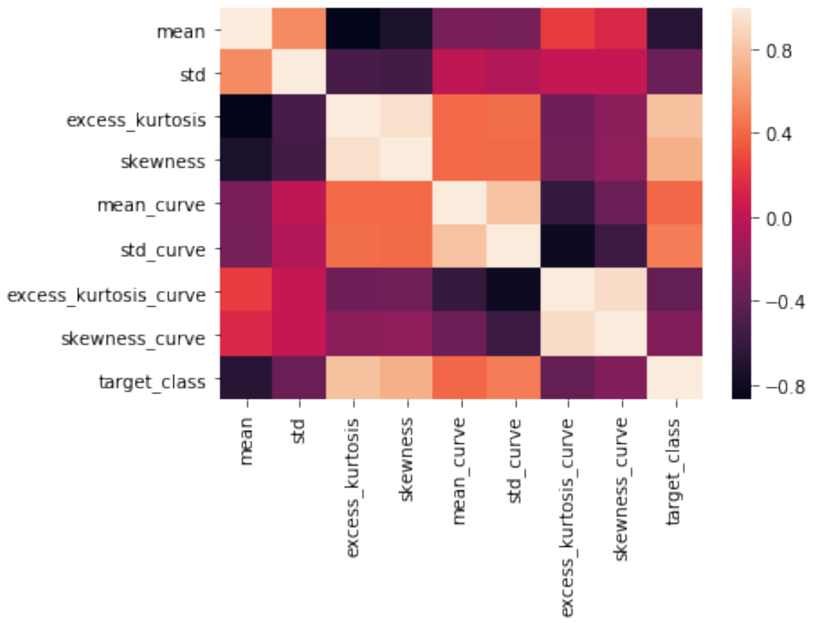


We can see from the above plot, that only `skewness` and `excess kurtosis` has a fair degree of linear correlation. Rest of the pairs don't share such relations.



In the above plot, the *skewness* feature has the maximum skewness out of all features.

The following heatmap shows the linear correlation between every pair of variables, which we had seen before from the scatter plot.



Algorithms and Techniques

We'll use 4 different classifiers.

1. Logistic Regression
2. Support Vector Machines (SVM)
3. Ensemble Method (AdaBoost)
4. XGBoost

We'll be tuning the relevant hyperparameters for each of these classifiers. We'll use 5-fold cross validation. So, our data will be partitioned into 5 different buckets, and for each iteration, 4 of these shall be used as training set, and the remaining one will be used as testing set. We'll get average train and test score for this whole process. Now, as we are tuning the hyper-parameters, we'll get a grid of those parameters, and for each entry we'll be running the 5-fold cross validation. In the end we'll choose those hyperparameter for which our F-beta score will be maximum.

Benchmark

As this is a classification problem, we can keep a naïve predictor (which considers all starts as non-pulsar) as a benchmark. This is because, almost 90% of our dataset consists of non-pulsars. We'll then compare our accuracy, precision/recall against that one. As we don’t want to lose out on actual pulsars, so we’d be giving more emphasis on recall than precision.

III. Methodology

Data Pre-processing

1. From initial observation, the dataset seems to be clean, so I presume there won’t be much need for data sanitization. We can directly use this data for exploration and visualization.

2. It is often good practice to perform normalization on numerical features, as it doesn't alter the feature's distribution and also ensures each feature is treated equally when applying supervised learners. We will use `sklearn.preprocessing.MinMaxScaler` for normalization.

Naive Predictor Performance

We're blindly saying all stars are pulsar. Here we're using beta = 2 because we care more about recall than precision. It gives the following performance metrics. We could have chosen the naïve predictor for non-pulsars, but in that case, it would be precision heavy, and we would've needed to put the beta as a fraction between 0 and 1. For consistency, we keep it in this way.

```
Naive Predictor: [Accuracy score: 0.0916, Precision: 0.0916, Recall: 1.0000,
F-score: 0.3351]
```

Implementation

Models of choice: Logistic Regression, SVM, AdaBoost and XGBoost.

Logistic Regression

- Real world application:
    1. Image Segmentation and Categorization
    2. Geographic Image Processing
    3. Handwriting recognition
- Strength:
    1. The decision boundary it creates is a linear decision boundary that can be of any direction. So, if we have data where the decision boundary is not parallel to the axes, then logistic regression picks it out pretty well, whereas a decision tree will have problems.
    2. Logistic regression is intrinsically simple, it has low variance and so is less prone to over-fitting.
- Weakness:
    1. Logistic regression attempts to predict outcomes based on a set of independent variables, but if we include the wrong independent variables, the model will have little to no predictive value.
    2. Logistic regression works well for predicting categorical outcomes like admission or rejection at a particular college, but it cannot predict continuous outcomes.
    3. Logistic regression requires that each data point be independent of all other data points. If observations are related to one another, then the model will tend to overweight the significance of those observations.
    4. Logistic regression attempts to predict outcomes based on a set of independent variables, but logit models are vulnerable to overconfidence. source
- Why This model:
    - It's a good place to start with, because it's a simple baseline algorithm for binary classification, and it performs reasonably well even with its assumptions of independence.

SVM

- Real world application: Classification of images. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. source
- Strength:
  1. It has a regularisation parameter, which makes the user think about avoiding over-fitting.
  2. It uses the kernel trick, so we can build in expert knowledge about the problem via engineering the kernel.
  3. An SVM is defined by a convex optimisation problem (no local minima) for which there are efficient methods (e.g. SMO).
  4. Lastly, it is an approximation to a bound on the test error rate, and there is a substantial body of theory behind it which suggests it should be a good idea. source
- Weakness:
  1. Requires full labeling of input data
  2. Uncalibrated class membership probabilities -- SVM stems from Vapnik's theory which avoids estimating probabilities on finite data
  3. The SVM is only directly applicable for two-class tasks. Therefore, algorithms that reduce the multi-class task to several binary problems have to be applied.
  4. Parameters of a solved model are difficult to interpret.
  5. Takes quite a while to learn, on a large dataset. source
- Why This model:
  - Tried and tested algorithm for binary classification. Kernel trick can be employed to capture more complex relationships.

Ensemble Methods (Adaboost) source

- Real world application:
  1. Face detection (Complex boundary detection)
  2. Image Processing
- Strength:
  1. It's fast and adaptive.
  2. Less susceptible to overfitting.
  3. It can be used in conjunction with many other types of learning algorithms to improve performance.
- Weakness:
  1. Sensitive to noisy data, and outliers.
- Why This model:
  - It's a good model as our dataset is not that complex, and it usually gives more accuracy than simpler classification models (e.g. Logistic Regression)

XGBoost source

- Real world application:
  1. Any production systems using GBM can use XGBoost.
- Strength:
  1. Provides scalable, portable and distributed Gradient Boosting.

    2. it provides state-of-the-art performance for typical supervised machine learning problems.
- Weakness:
    1. It cannot handle categorical features by itself, it only accepts numerical values similar to Random Forest.
- Why this model:
    - It usually beats the other boosting algorithms by a fair margin. It was the algorithm of choice for many winning teams of a number of machine learning competitions (e.g. Kaggle)

How do we proceed

1. We use K-fold cross validation with K=5. As our target data set is imbalanced (about 9% of stars in the data set are pulsars) we need to preserve a similar ratio in our train-test data sets. This is handled by K-fold cross validation as by default it uses stratified sampling.
2. For each of the classifiers we identified a parameter grid, which we are going to tune. For each combination of these hyper-parameters we run a K-fold cross validation and note down the average F-beta score.
3. Whichever hyperparameters give the best score, are extracted out as the optimal estimator.
4. We compare all 4 optimal models and choose one that gives us the best F-beta score.

Refinement

The refinement process is embedded in the grid-search procedure, as it helps us finding the best set of hyper parameters.

IV. Results

Model Evaluation and Validation

I think, XGBoost is the most appropriate model, for identifying pulsars. The reasons being:

1. It has highest F-score across all proportions of testing sets.
2. It has highest accuracy across all proportions of testing sets.
3. SVM takes significant time to train, compared to XGBoost or AdaBoost, while giving slightly less F-score and accuracy.
4. Compared to Logistic Regression, the accuracy and F-scores are much better. It can also handle more complex relations among data points, compared to Logistic Regression.

Also, we used K-fold cross validation while training our model, so it generalizes much better than the situation, when one picks up a random train-test split. Because in that case, this model might have become lucky. But since, we've tested over a normalized set of train-test pairs, this model is much likely be better than the other 3 for unknown data sets too.

Justification

The XGBoost F-beta score is way higher than the naive predictor which is expected. Compared to the other three model, it dominates the field in all accuracy, precision, recall and F-beta score, so it's a clear winner among all the models we've chosen to experiment with. The naive predictor has a recall score of 1, because it blindly classifies everything as pulsar, but it's low precision score brings down the F-beta score. In XGBoost, we have a good balance between both precision and recall.

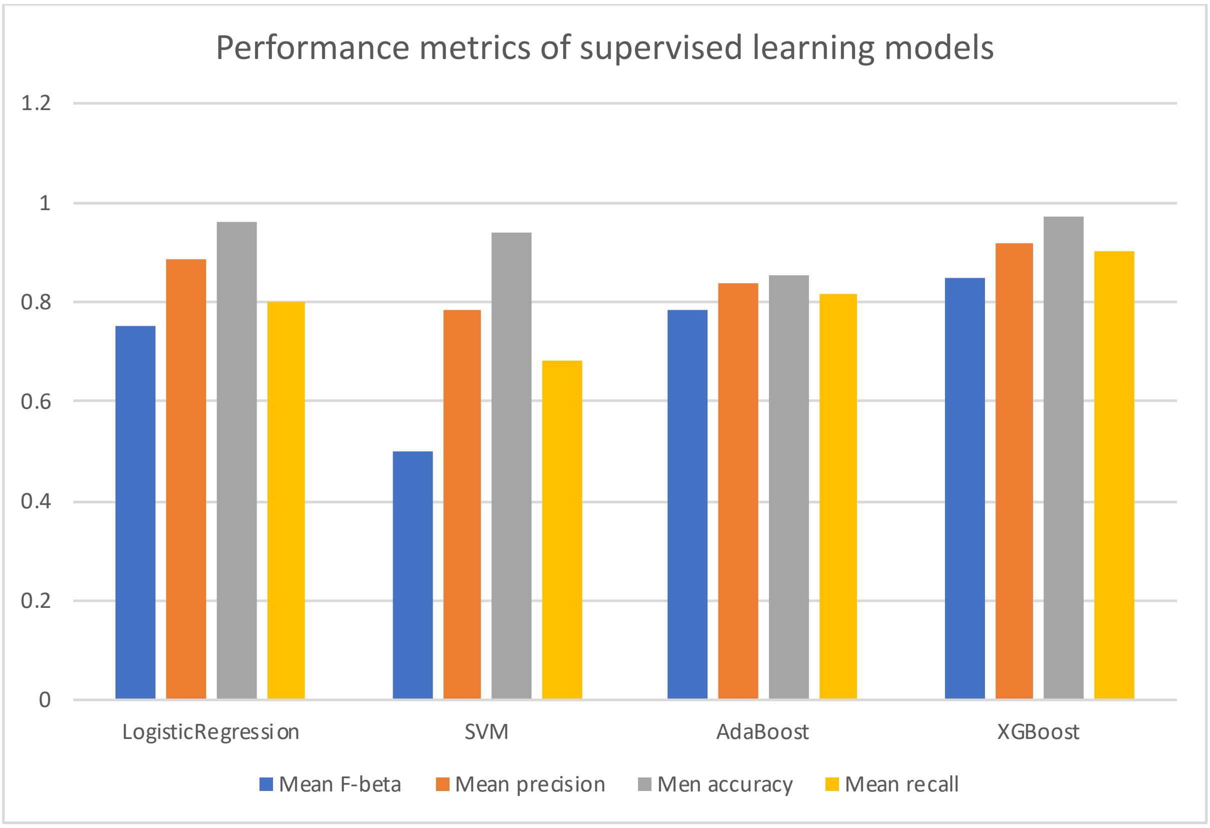
V. Conclusion

Free form visualization

The final performance metrics that we obtained from our models are shown below:

```
[
	{
		'model': 'LogisticRegression',
		'mean_f-beta': '0.7512689089358641',
		'mean_precision': '0.8889054836876458',
		'mean_acc': '0.9607298501029644',
		'mean_recall': '0.8011004857332923'
	},
	{
		'model': 'SVM',
		'mean_f-beta': '0.4976414958449544',
		'mean_precision': '0.7857228934604635',
		'mean_acc': '0.9408202033746788',
		'mean_recall': '0.6839523686289998'
	},
	{
		'model': 'AdaBoost',
		'mean_f-beta': '0.7837768512211324',
		'mean_precision': '0.8364843142824105',
		'mean_acc': '0.8515024291088654',
		'mean_recall': '0.8141367209627456'
	},
	{
		'model': 'XGBoost',
		'mean_f-beta': '0.84731026071335',
		'mean_precision': '0.9193869932426746',
		'mean_acc': '0.971542444220956',
		'mean_recall': '0.9005733520660092'
	}
]
```

Let’s visualize it in the following way.



This clearly shows that XGBoost outperforms all other models in our use case.

Reflection

This was quite an interesting project to work on, as it gave me some new challenges to solve and reinforced some of my understanding regarding classifiers. To summarize, our input was a dataset with 8 input features and one target feature. The goal was to correctly classify objects as pulsar or non-pulsar. We used sklearn's gridsearchCV along with k-fold cross validation to find the optimal parameters of each model, and then choose the best model out of all four (Logistic Regression, SVM, AdaBoost and XGBoost). We found XGBoost was the best in class as it had the best F-beta score. This was kind of expected as this is one of the heavily used supervised learning models used in Kaggle. I struggled a bit on combining both grid search and k-fold cross validation in sklearn, but hopefully that got resolved.

Improvement

I think there are quite a bit of room for improvement. E.g.

1. I'm not much familiar with stacking multiple models and create a custom ensemble model which tend to give better performance in general than the constituent models. I couldn't try that out because of my lack of knowledge in that front.
2. Almost each of these models provide a myriad of hyperparameters to tune, which can become overwhelming sometimes. From my limited knowledge in this area, I've

chosen minimal set of parameters to tune, but obviously there're lot of room for improvement here.

3. Even if 2-3 features where heavily skewed, I didn't do a log transformation on those, to keep it simple. The log transformation might have reduced the adverse effect of outliers (if any).