Capstone Project

Machine Learning Engineer Nanodegree

Andrea Balzano

19th April 2017

The Search for New Earths

Definition

Project overview

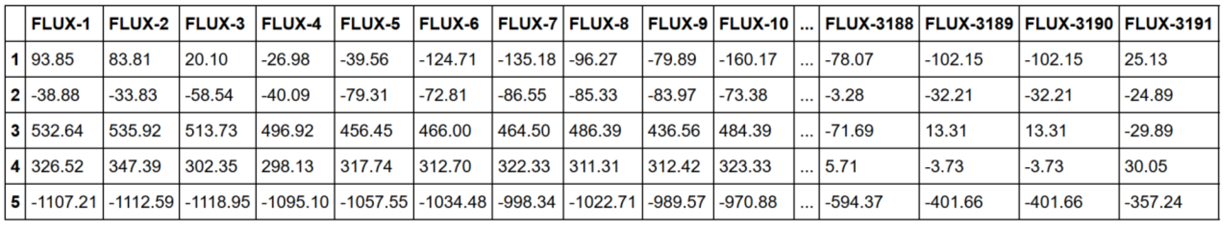
This project analyses the data observed by the NASA Kepler space telescope searching for exoplanets using the transit technique.

planets themselves do not emit light, but the stars that they orbit do. If said star is watched over several months or years, there may be a regular 'dimming' of the flux (the light intensity). This is evidence that there may be an orbiting body around the star; such a star could be considered to be a 'candidate' system.

NASA itself utilises python to interpret the data and has created PyKE, a library for data reduction to help with extraction and preprocessing of the light curve images, however this project analyses only FLUX data, not pictures.

This is an example of the data for 5 known candidate systems, the luminescence is represented with a float number, and in this dataset, it ranges from -6.724046e+05 to 3.616292e+06.

Each column is a Snapshot of the luminescence at a fixed time interval:



	FLUX-1	FLUX-2	FLUX-3	FLUX-4	FLUX-5	FLUX-6	FLUX-7	FLUX-8	FLUX-9	FLUX-10	...	FLUX-3188	FLUX-3189	FLUX-3190	FLUX-3191
1	93.85	83.81	20.10	-26.98	-39.56	-124.71	-135.18	-96.27	-79.89	-160.17	...	-78.07	-102.15	-102.15	25.13
2	-38.88	-33.83	-58.54	-40.09	-79.31	-72.81	-86.55	-85.33	-83.97	-73.38	...	-3.28	-32.21	-32.21	-24.89
3	532.64	535.92	513.73	496.92	456.45	466.00	464.50	486.39	436.56	484.39	...	-71.69	13.31	13.31	-29.89
4	326.52	347.39	302.35	298.13	317.74	312.70	322.33	311.31	312.42	323.33	...	5.71	-3.73	-3.73	30.05
5	-1107.21	-1112.59	-1118.95	-1095.10	-1057.55	-1034.48	-998.34	-1022.71	-989.57	-970.88	...	-594.37	-401.66	-401.66	-357.24

Some of the machine learning techniques already been used by developers are 1-D CNN, XGBoosting, PCA.

Known Dataset problem

The Transit technique has some flaws, it can be used to investigate only stars lying on the same hyperplane as the sun and it can detect only planets that orbit their stars with a very short period because it needs to be able to see more than one orbit to be able to differentiate between a transit and background noise.

For example, looking at our own solar system, to be able to confirm the existence of the Earth we would have to constantly observe the sun for at least 2 years to see 2 transits and it probably wouldn't be enough information to differentiate the dip in light caused by the Earth because of the possibility of dips caused other bodies.

To identify Jupiter Following this technique we would have to observe the sun for 24 years just to see 2 transits and for some outer planets like Saturn and Pluto respectively 59 and 496 years.

Source: Nasa

For this project, it means that observation marked as non-exoplanet systems might have planets, creating extra noise and further complicating the classification task.

Problem Statement

The goal is to create an agent able to classify candidate systems.
At the time this dataset was prepared Campaign-3 was unlikely to contain any undiscovered exoplanet-stars. Therefore, all stars (i.e. all rows) which were not confirmed to host an exoplanet, were labelled with a 0. This is over 99% of the observations.
In total, there are 42 observations of confirmed exoplanets labelled with a 1, 5 in the test set and 37 in the train set.
The task involved are the following:

1 Download the dataset.
2 remove outliers and normalise the data.
3 correct for class imbalance
4 run preprocessing analysis.
5 run classification algorithm.

The output of the classifier is a binary class where a system not containing exoplanets is labelled 0, and a candidate for more investigations is labelled 1.

Metrics

class 0 includes 99% of samples, due to this high class label imbalance, the metric used will be the sensitivity and specificity scores.

Sensitivity (also called the true positive rate) measures the proportion of positives that are correctly identified as such.
i.e. the percentage of candidate systems which are correctly identified as candidate.

Specificity (also called the true negative rate) measures the proportion of negatives that are correctly identified as such.
i.e. the percentage of systems which are correctly identified as not being candidate.

This metric was used to evaluate the classifier because it best represents false negative and false positive when high class imbalance.

- False negatives result in a missed opportunity, these are systems that actually have exoplanets but are not classified as such.
- False positives result in further investment of resources to study in details the system only to find out that we can't identify any exoplanets.

Analysis

Data exploration

the Trainset includes:

- 5087 rows or observations.
- 3198 columns or features.
- Column 1 is the label vector. Columns 2 - 3198 are the flux values over time.
- 37 confirmed exoplanet-stars and 5050 non-exoplanet-stars.

The Testset includes:

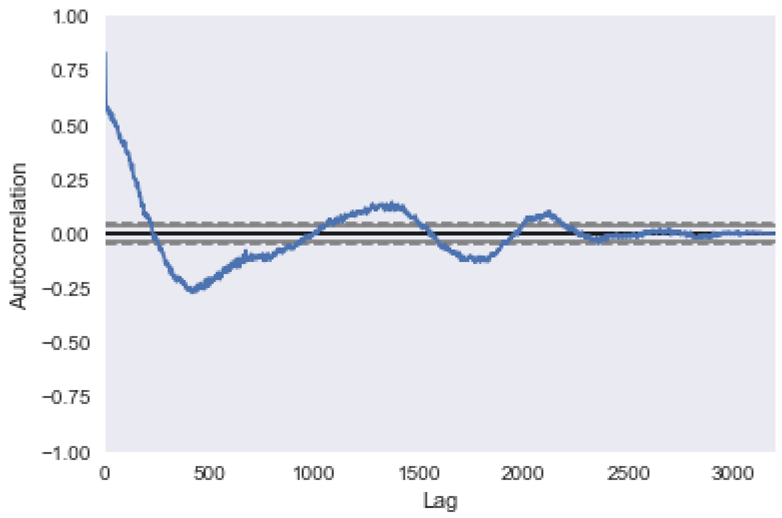
- 570 rows or observations.
- 3198 columns or features.
- Column 1 is the label vector. Columns 2 - 3198 are the flux values over time.
- 5 confirmed exoplanet-stars and 565 non-exoplanet-stars.

some descriptive statistics on the data:

	FLUX-1	FLUX-2	FLUX-3	FLUX-4	FLUX-5
count	5.087000e+03	5.087000e+03	5.087000e+03	5.087000e+03	5.087000e+03
mean	1.445054e+02	1.285778e+02	1.471348e+02	1.561512e+02	1.561477e+02
std	2.150669e+04	2.179717e+04	2.191309e+04	2.223366e+04	2.308448e+04
min	-2.278563e+05	-3.154408e+05	-2.840018e+05	-2.340069e+05	-4.231956e+05
25%	-4.234000e+01	-3.952000e+01	-3.850500e+01	-3.505000e+01	-3.195500e+01
50%	-7.100000e-01	-8.900000e-01	-7.400000e-01	-4.000000e-01	-6.100000e-01
75%	4.825500e+01	4.428500e+01	4.232500e+01	3.976500e+01	3.975000e+01
max	1.439240e+06	1.453319e+06	1.468429e+06	1.495750e+06	1.510937e+06

Being the features a sequence of the same measurement we can visualise the autocorrelation, here on a sample candidate system

Fig1: Shows the autocorrelation between FLUX



Exploratory Visualisation

Fig 2: Shows how the luminescence dims when an exoplanet transit in front of the star for the 37 known exoplanets in the trainingset, in these cases we can observe system with multiple, short, or long transits:

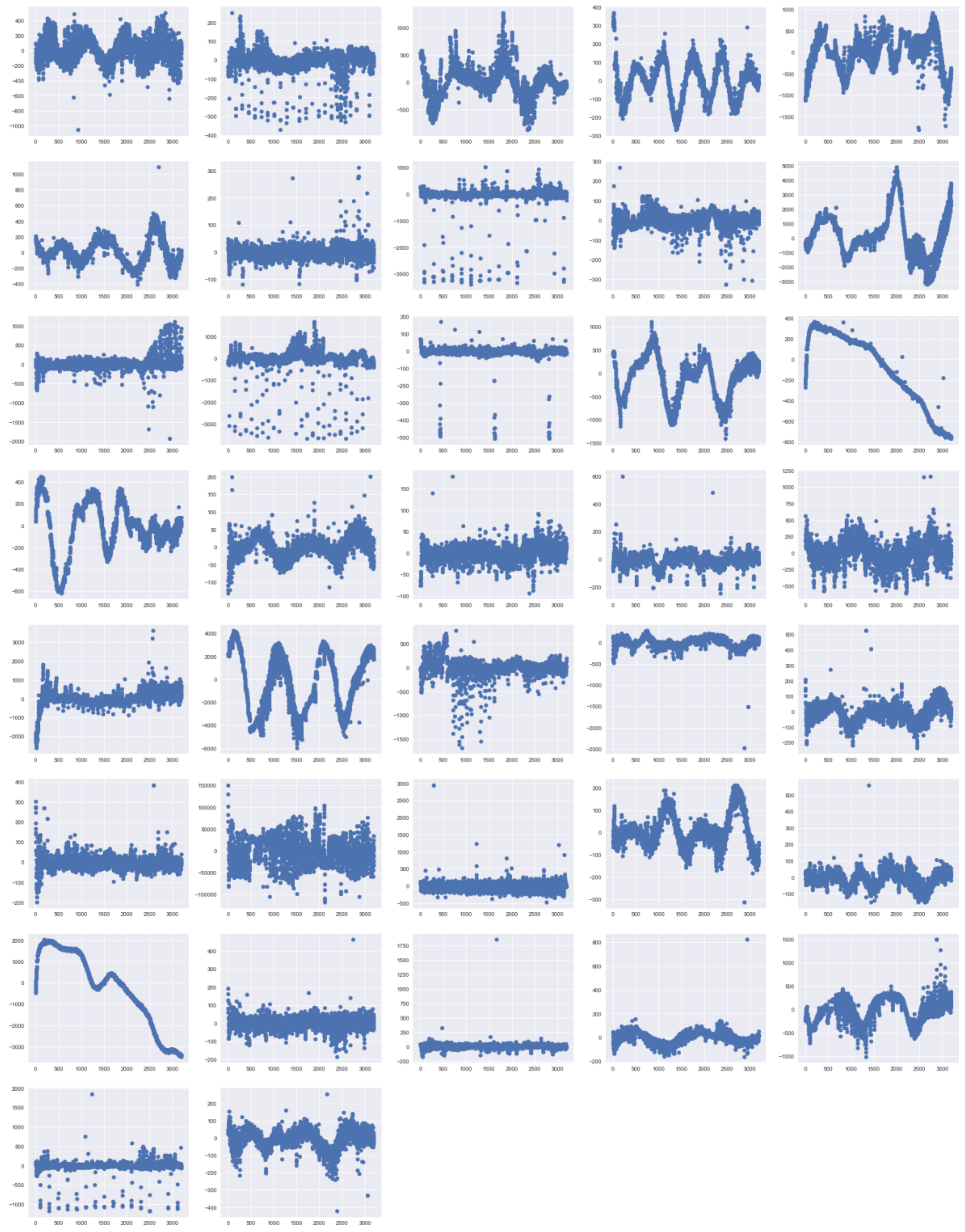


Fig 3: shows outliers for class 0, non-exoplanets systems, some of these observation presents strong sinusoidal noise or high fluctuations in short times that might be caused by interference with instruments, others might actually show one transit of a planet that we are not yet able to identity.

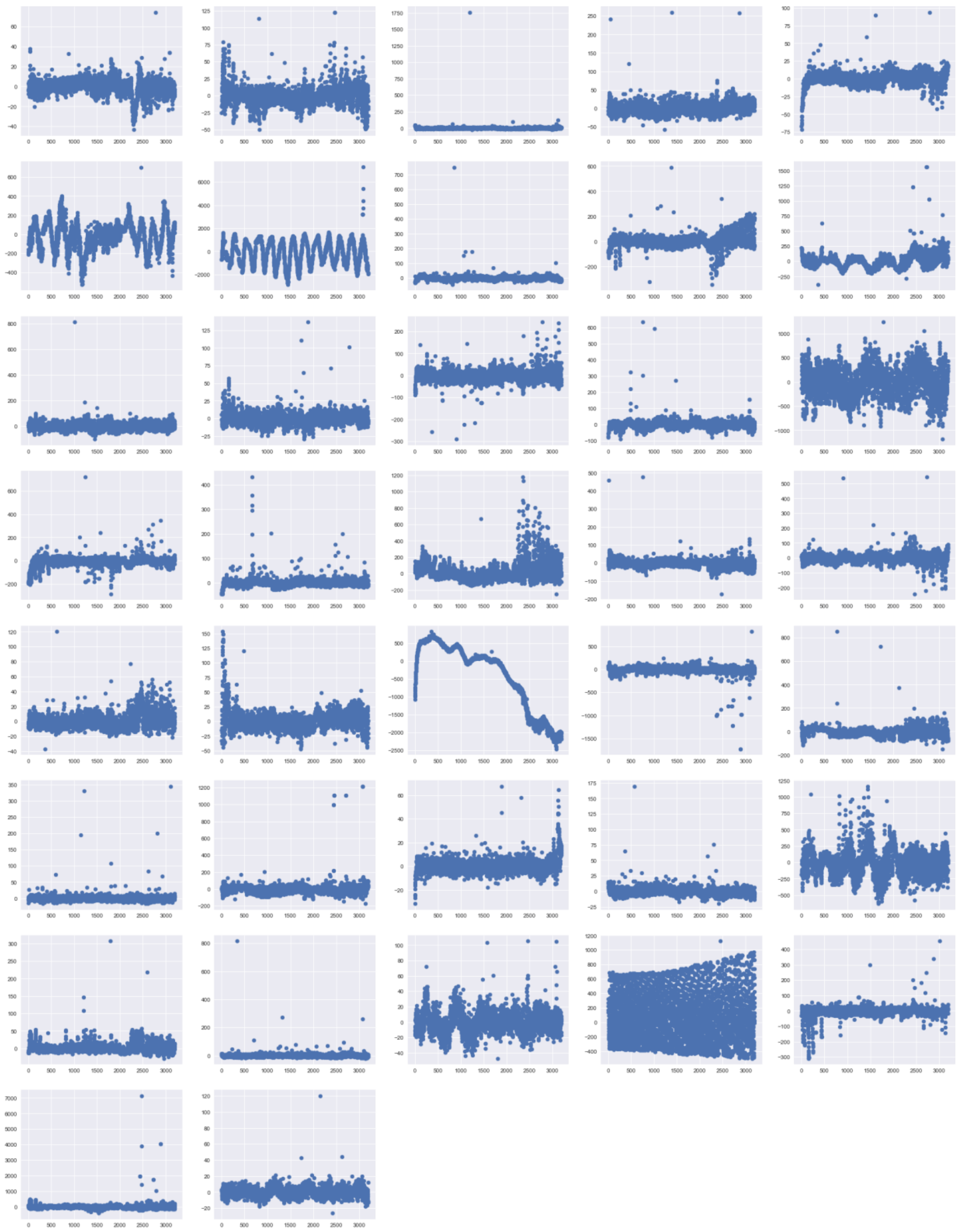
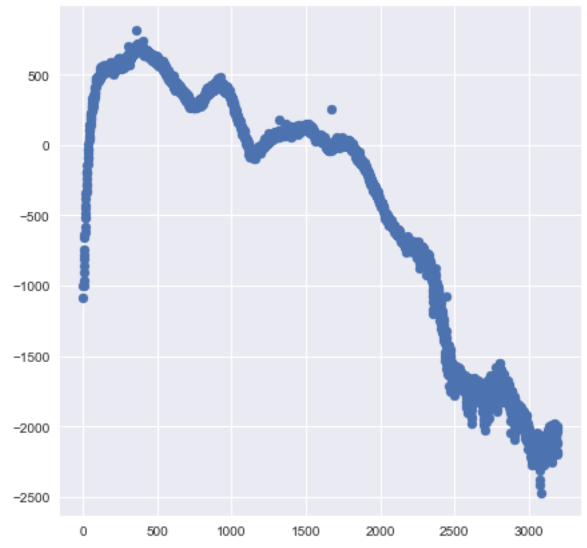


Fig 4: this particular outlier could be showing the transit of a slow-moving gas giant planet.



Algorithms and Techniques

I explored multiple techniques in this project, some more successful than others.

Initially considering that This dataset has very high dimensionality represented by the time series of the FLUX measurements, I did analyse the data looking for outliers that can be due to noise or sensors anomalies, approaching the detection using Tukey's Method for identifying outliers, where an outlier step is calculated as 1.5 times the interquartile range (IQR).

I Choose his method because it isn't dependent on distributional assumptions, and It also ignores the mean and standard deviation, making it resistant to being influenced by the extreme values in the range.

I filtered all the stars with measurements under/over the outlier step looking for periodicity on these values to try isolate noise and decide on a case basis if to keep the record, remove it or replace the value of the outlier with the median value for that column.

I addressed the class imbalance by applying SMOTE + Tomek links to try to balance the tradeoff of over/undersampling.

The SMOTE technique increases the size of the minority class by creating samples in the neighbors, correcting for the tendency to overfit to the samples common during oversampling.

The Tomek links is a redundancy driven technique to reduce sampling size, it looks for Tomek-links which consist of points that are each other's closest neighbors, but do not share the same class label.

For a binary class, the number of possible combinations grows exponentially with the number of dimensions, 2^n where n is the number of features, this is also known as the curse of dimensionality,

To address this I applied Principal Component Analysis, this is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components (or sometimes, principal modes of variation). The number of principal components is less than or equal to the smaller of the number of original variables or the number of observations.

To classify the data I did use Gradient Boosting, in this particular implementation of the algorithm the classification is binary and in these cases, it uses one regression tree fit on the negative gradient of the binomial or multinomial deviance loss function.

The drawback of this model is that it can quickly fit, then overfit the training dataset.

The Learning rate is used to slow down the learning in the gradient boosting model by applying a weighting factor for the corrections by new trees when added to the model, generally, it is used with a small value like 0.1.

The number of estimators is the number of the total tree that will be added to the model, increasing the number help to better fit the data but it can quickly lead to overfitting.

Minimum impurity split is the Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise, it is a leaf. This hyperparameter controls how big each estimator can be, tuning this value will have effects on overfitting.

The max depth is the max number of branches each estimator can have, as per the minimum impurity, tuning this value will have effects on overfitting.

The one Class Classifier for outlier detection used is the Isolation Forest, this algorithm isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

The logic arguments go: isolating anomaly observations is easier as only a few conditions are needed to separate those cases from the normal observations. On the other hand, isolating normal observations require more conditions. Therefore, an anomaly score can be calculated as the number of conditions required to separate a given observation.

The way that the algorithm constructs the separation is by first creating random decision trees. Then, the score is calculated as the path length to isolate the observation.

In order to avoid issues due to the randomness of the tree algorithm, the process is repeated several times, and the average path length is calculated and normalized.

This random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produce shorter path lengths for particular samples, they are highly likely to be anomalies.

Benchmark

Since this is a research project, there are not real clear benchmarks available yet.

However, I am expecting that the best algorithm will have an average precision metric of at least 10% considering that a naïve Bayes classifier trained on the raw data even though it averages the precision to 30% on the validation set, it classifies always as class 0 on the test set.

Methodology

Data preprocessing

The preprocessing done in the "prepare data" section of the notebook consists of the following steps:

1. The author labelled the classed as 1 and 2, I will change it to 0 and 1 more machine learning friendly.
2. Normalising the data by sample, in this dataset, all dimensions are the same feature, therefore, normalizing by feature wouldn't be useful while normalising each sample can help identifying the dips.
3. outliers detection using Tukey's Method for identifying outliers.
4. Class imbalance correction by applying SMOTE + Tomek links.
5. Features transformation with PCA.

Implementation

The implementation process in done in a notebook with the use of a helper file for functions to simplify the use of the notebook.

The steps are:

1. Load the data and preprocessing as above.
2. Implement helper functions:
    a. find_outliers() returns an object counter with the indexes and counts of the outliers
    b. fit_model() returns a classifier object trained with gridsearch.
3. Defined feature selection, classifier and training parameters.
4. Imported metrics function.
5. Trained classifier, validation.
6. If sensitivity not high enough return to step 3.
7. Save the trained classifier.

I did apply features transformation with PCA to capture the delta changes over time, reducing the number of features and compensating for the curse of dimensionality.

I did try Gaussian Naïve Bayes, SVM, Gradient boosting classifiers to get a benchmark, and while all classifier tested had a strong tendency to overfit, I explored the latest algorithm to fine tune it using cross-validation on the training set.

Ones the score on the validation set was acceptable I did evaluate the performance on the testset.

I noticed that without normalising the data PCA was very effective in reducing dimensionality while retaining information, with 2 features it could explain 83% of the variance and 99% with just 10.

After applying normalisation, normalising by sample or using standard scaler the PCA was little effective and I had to use 30 - 50 features to explain just 45% of the variance.

The results of the test set following this solution were always biased towards one of the two classes, the best classifier had about a sensitivity of 80% and specificity of 7%.

I investigated the possibility of doing a time series analysis but the algorithms are designed for problems with variable outcome over time while this problem is about an overtime change that leads to one outcome.

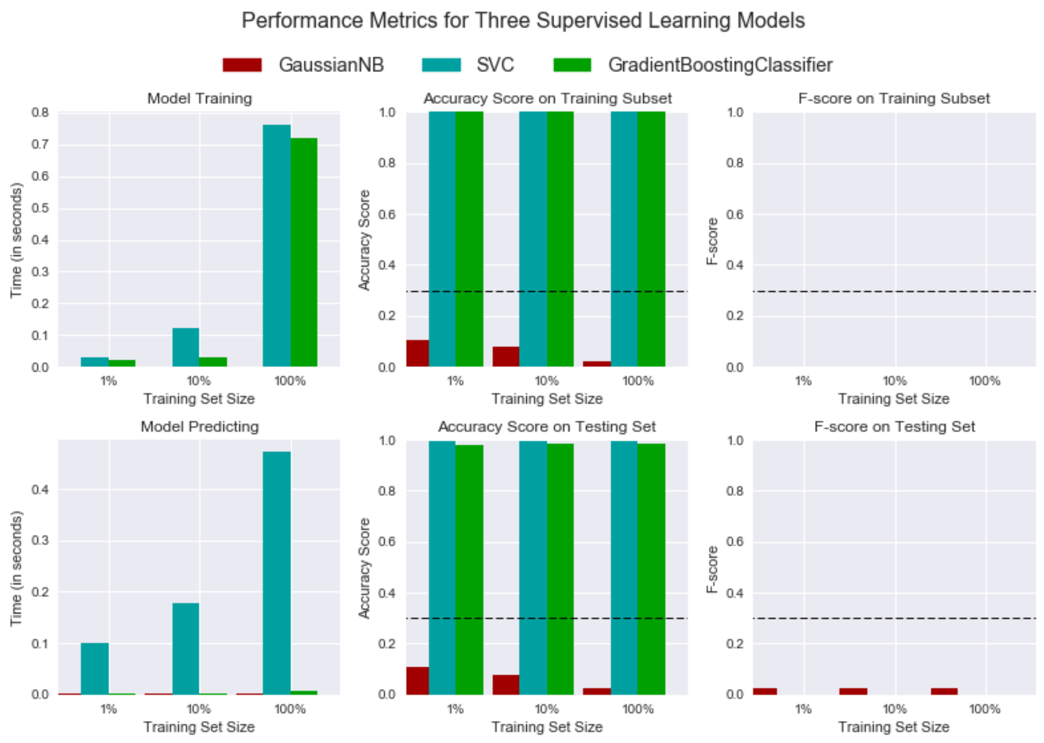
Since then I moved to one class classifier for outlier detection, I approached the problem differently by not applying outlier removal or class imbalance correction, training only on class 0.

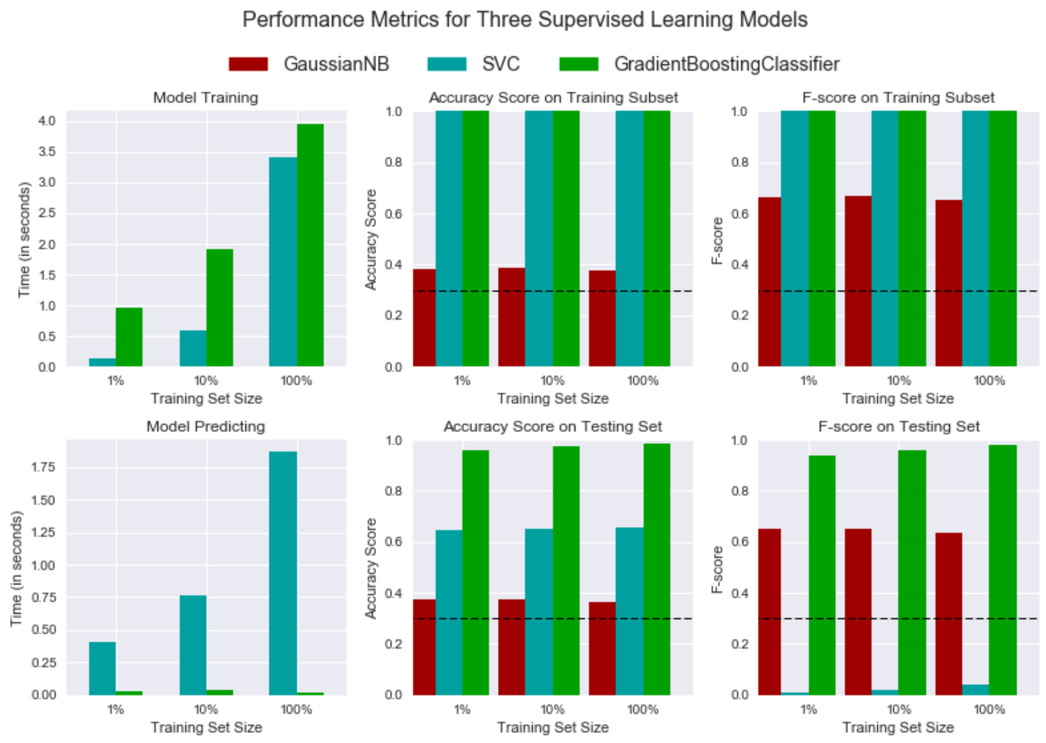
The algorithms I used are one class SVM and isolation forest, the later was the final choice because being able to better interpret the variation on dip sizes and frequency.

The most challenging part of the code was working with One Class Classifier, a topic that was new to me, I spend most of the time trying to visualise decision boundaries to be able to understand the sanity of the models and to help me chose the best one.

Refinements

Without correction for class imbalance the classifiers are biased and overfit to class 0 with a poor F1 score, applying outlier removal and SMOTE mitigated the issue.





Using grid search and cross validation I tested multiple hyperparameters:

Gradient boosting:

- learning_rate: (0.1,0.08,0.01,0.5)
- n_estimators: (50,100,180,200,250)
- max_depth: (3,4,5,6,7,8,10)
- min_impurity_split: (10e5,10e6,10e7)

PCA:

- n_components: (5,10,40,44,45,46,50,80,100)

The final parameters used are:

Gradient boosting:

- n_estimators = 100
- learning_rate: 0.1
- max_depth= 7
- min_impurity_split= 100000.0

PCA:

- n_components = 46

Results

Model Evaluation and Validation

During development, a validation set extracted from the trainset was used,

When experimenting with one class novelty and outlier detection the full training set of class 0 was used for training and the full trainset for validation.

Preliminary results with tuned gradient boosting on validation set:

```
                   pre       rec       spe        f1       geo       iba       sup

          0       1.00      0.92      0.72      0.95      0.31      0.11      2000
          1       0.10      0.72      0.92      0.17      0.31      0.09        25

avg / total       0.99      0.91      0.72      0.94      0.31      0.10      2025
```

Validation After correction for class imbalance:

```
                  pre       rec       spe        f1       geo       iba       sup

          0       0.99      0.99      0.98      0.99      0.98      0.97      2000
          1       0.98      0.98      0.99      0.98      0.98      0.97      1107

avg / total       0.98      0.98      0.98      0.98      0.98      0.97      3107
```

Preliminary results with One class SVM as outlier detection:

```
sensitivity: [ 0.9998  0.    ], specitivity: [ 0.      0.9998], support : [5000   37]
```

Final results using Isolation Forest:

```
                  pre       rec       spe        f1       geo       iba       sup

          0       0.99      0.10      0.81      0.18      0.08      0.01      5000
          1       0.01      0.81      0.10      0.01      0.08      0.01        37

avg / total       0.98      0.11      0.81      0.18      0.08      0.01      5037
```

Results on test set:

```
                  pre       rec       spe        f1       geo       iba       sup

          0       0.98      0.10      0.80      0.18      0.09      0.01       565
          1       0.01      0.80      0.10      0.02      0.09      0.01         5

avg / total       0.97      0.10      0.79      0.18      0.09      0.01       570
```

At the beginning of the project, I was expecting the Gradient Boosting to perform better on the test set while I was surprised at how stable are the results from the Isolation Forest, the metric scores are very similar between validation and test set proving it to be robust.

Justification

Comparing the Isolation Forest to the Naïve bayes does not have statistical meaning considering the naïve classifier was always predicting class 0.

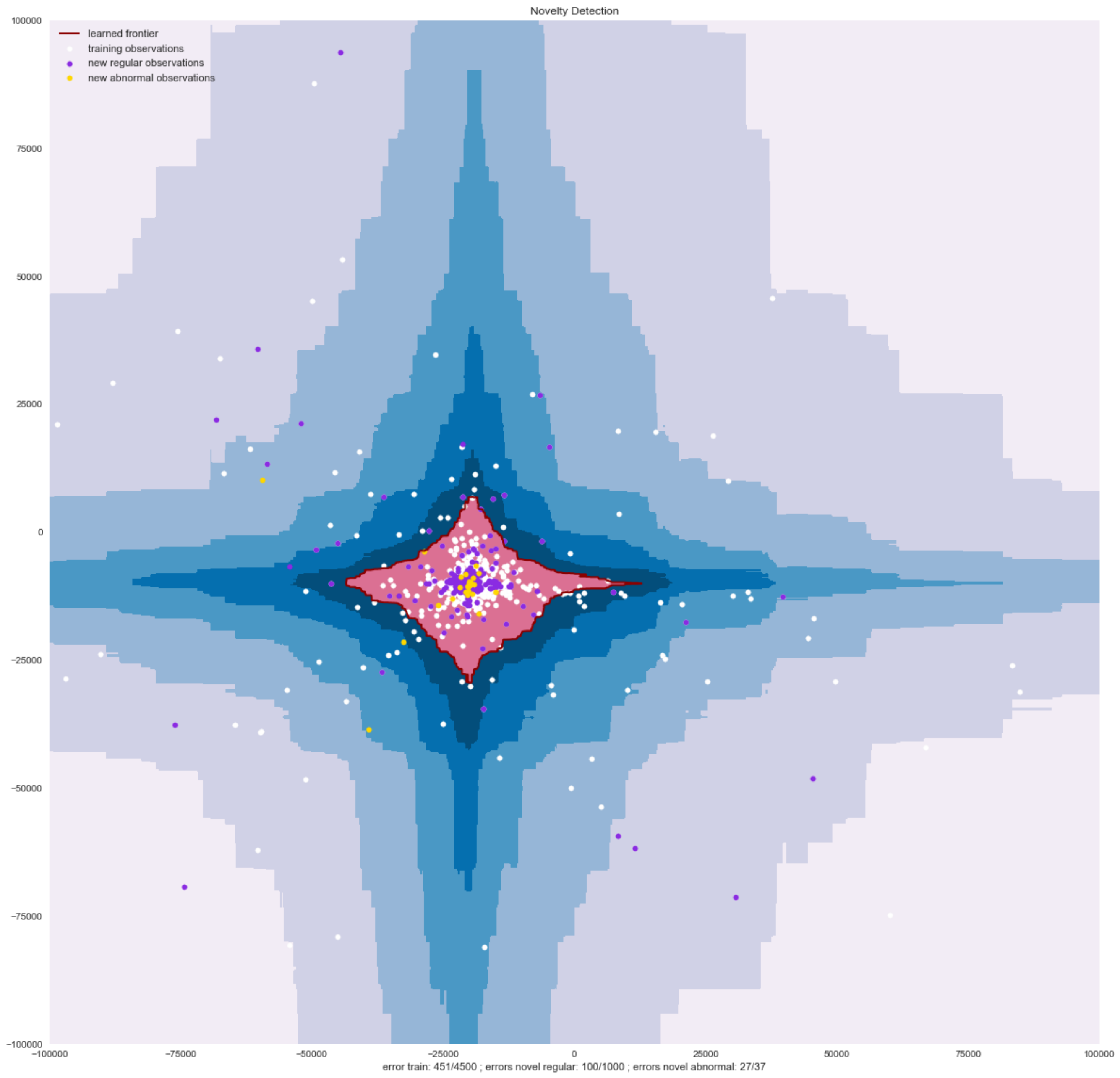
the result on the test set shows strong results for class 0 precision meaning it is not missing out on finding new planets and the 80% recall score on class 1 shows it is good at finding the candidates, however there is a big amount of false positive that would have to be further scrutinized, I believe this is a good result because it can be used with success in eliminating systems where we can't observe planets and it could be used as first screen for new data, accelerating the initial analysis. To better solve this problem other classification techniques can be used to improve the false positive rate. (see the Improvement section)

Conclusion

Free Form Visualisation

Fig 4: in this graph, we can visualise the decision boundaries for the isolation forest, where inside the red area the data is classified as class 0 and expanding out we can see how the probability of being class 1 increases.

This is to be considered for visualisation purposes only since the features have been reduced to 2, it does not reflect the actual accuracy of the final classifier.



Reflection

The process used for this project can be summarised using the following steps:

1. initial problem and relevant public dataset were found.
2. The data was downloaded and preprocessed
3. A benchmark was created for the classifier
4. Multiple classifiers were trained on the data to try solve the problem

I found that step 4 the most difficult, most of the common technique didn't produce any valuable results on validation test.

The most interesting part of the project was working with class imbalance and one-class classifiers, topics that were new to me.

Improvements

Different One class outlier detection methods can be tried and tuned to decrease the false positive rate giving better overall sensitivity.

Recursive neural networks are a type of classifier that can perform well on complex problems but it's tendency to overfit can be a problem considering this train data all algorithms had tend to overfit.