# Detection of Pokemons using Ensemble Methods

**Ted Scully**[1] **and Shrey Mishra**[2]

**Abstract:** In this paper, we will examine a range of different techniques that can be applied for the detection of anomalies on a relatively small Imbalanced dataset(with just 800 rows). We will also discuss a range of different evaluation techniques that can be used to evaluate the performance of such datasets.

## 1 Introduction

For the Demonstration of the project, we will use the Pokemon dataset available on Kaggle[1]. There are several Kernels submitted for the project but they are mainly highlighting the visualization aspect of the dataset and none of them go into a comprehensive evaluation of different Ml algorithms.

In this paper, we will try various optimal hyper-parameters that can be used together with a final Ensemble model that will take a majority soft vote in-order to finally classify the Legendary type Pokemons (Anomaly class).

In the dataset, there are nearly 8% (65) Legendary Pokemon's categorized as Anomalies that we are trying to detect. In our preprocessing, We will drop the feature "Pokedex Number" as it serves no meaningful insight. Also upon initial screening, we found that some Pokemons like "Charizard" have two different types "Fire" and "Flying," While some other Pokemons like "Pikachu" only have a single type i.e. "Electric". However, combining both the features we will have 18 unique categories that exist and all the Pokemons in this data can be classified within those 18 categories. To remove these categorical features we have performed the one-hot encoding method [2]. We have also replaced the labels of Legendary feature with class label 1 for True and class label 0 for False.

[1] Cork Institute of Technology
tedsculy@mycit.ie

[2] Cork Institute of Technology
shrey.mishra@mycit.ie

For the real-valued features such as Total, HP, Attack, Defense, Special Attack, Special Defence, Speed, and Generation we have performed standardization to scale each feature to unit variance. This is required for many algorithms such as PCA, KNN, logistic regression, etc,

In this paper, we will use a variety of supervised Machine Learning algorithms namely-

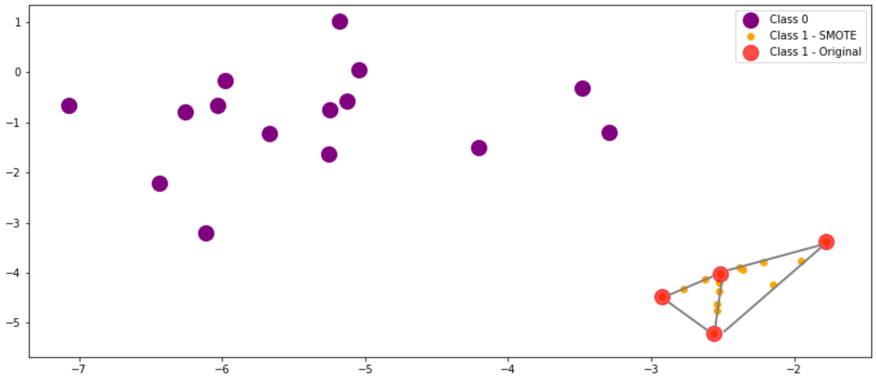
1.K- nearest Neighbours
2.Naive Bayes approaches
3.Logistic regression
4.Support Vector Machines based approaches
5.Decision Trees
6.Random Forest-based approach
7.Ensemble-based voting classifier

For visualization of the dataset, we will use techniques such as PCA and TSNE for the class imbalance we have tried various approaches using Smote and Upsampling.

## 2 Research

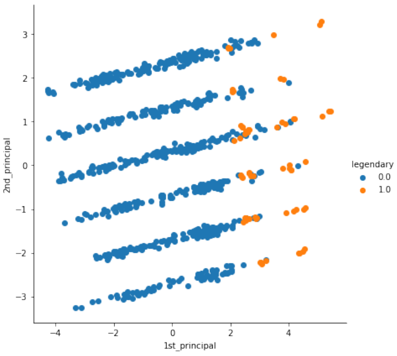
We have tried to deal with the class imbalance while using logistic regression techniques on class imbalance techniques such as the smote and the upsampling.

Using Smote, a technique that generates new fake samples to the anomaly class to deal with the class imbalance, where the new samples generated on the edges connecting the anomaly class meaning these new samples are generated on the hyperplane that connects the anomaly class[3]. As shown in the diagram below.

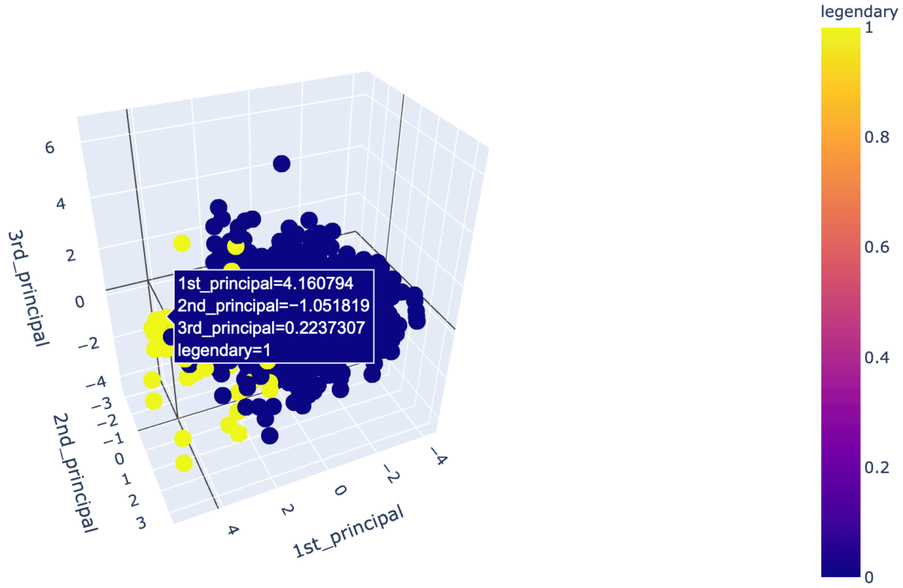


Using this technique however has its own disadvantages as when anomalies do not form a cluster formation, or some of them are completely apart from the dense cluster, the formation of new fake smote samples can cause a smote generated anomaly point to lie in the subspace of the inlier points.

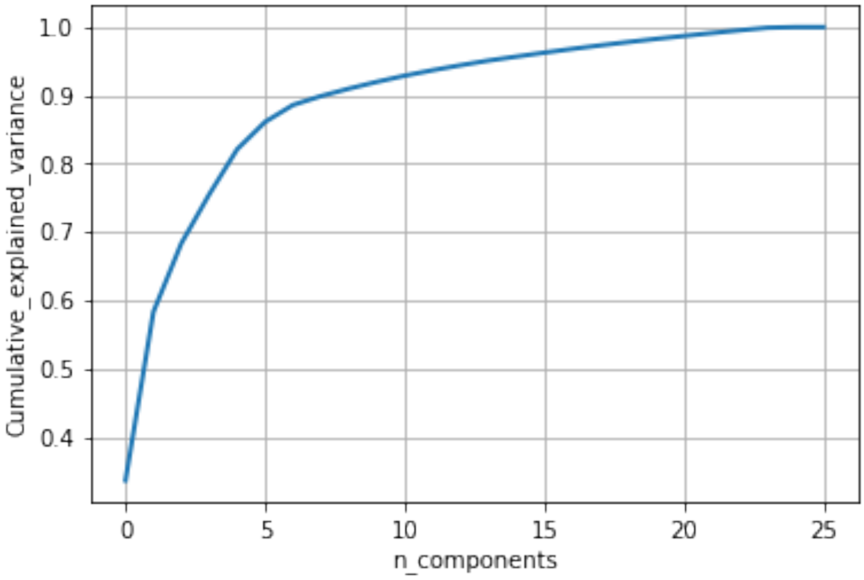
We will try to explain the concept using PCA[4] and 3d plots.



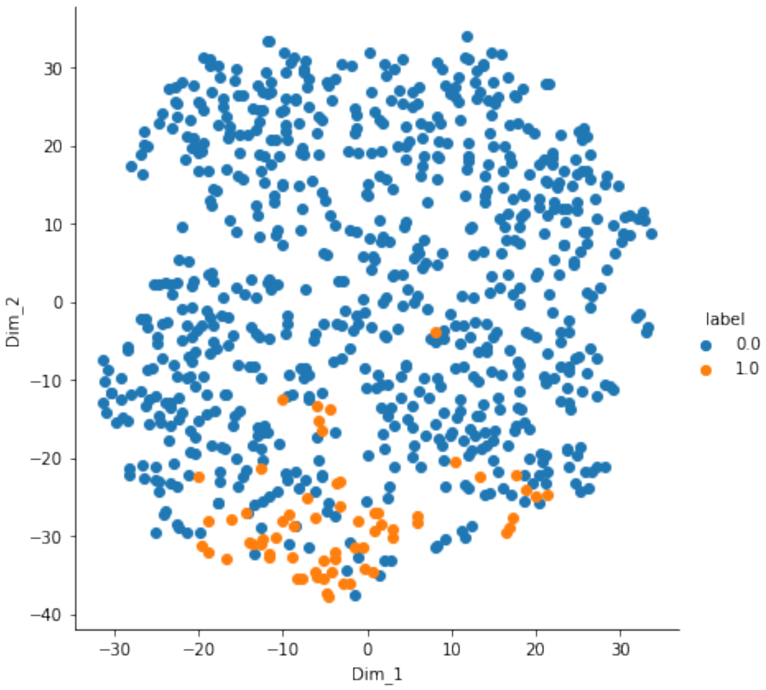
Below given is a 3D visualization of the 26-dimensional data

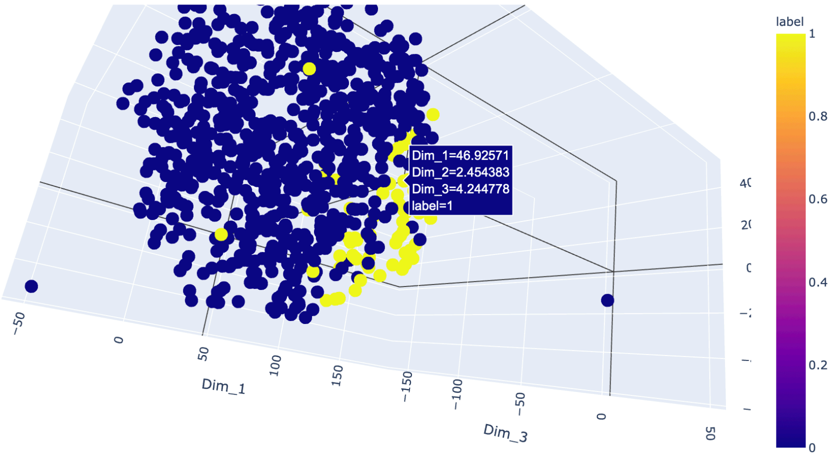


As we know that using PCA for dimensionality reduction would lose the variance from the data when we compress it to 3D as can be clearly explained with the variance chart given below.



To further improve the visualization we will imply a dimensionality technique called TSNE[5],





As we can see that there is a nice cluster formation of anomaly points, however, some of the anomaly points are very spaced from the dense cluster. Although Distances hold no meaning in the TSNE algorithm but still we can very safely assume that these points are not related to the dense cluster and thus generating smote samples would not help us. The use of smote to deal with Imbalance on Logistic regression helped us gain a very little increase in the 3 fold F1 score[6] from 18.88% to 29.52%. Thus an equally better way to deal is to assign more weights to the anomaly class which helped us reach the F1 score from the original 18.88% to 34% (approx) when assigning 1:9 ratio in favor of the Anomaly class.

When we implemented Upsampling we found an increased performance on a single run prediction on the test data, however, the 3 fold f1 score doesn't seem to improve.

We have done feature selection based upon the trained model from the logistic regression approach, where a higher weight corresponds to the more domination of a particular feature in deciding the class label of the data point. Whereas the negative point corresponds to the class it favors.

The top 5 features that decide the Pokemon is Legendary are all real-valued and not based upon the category of the Pokemon and are namely-

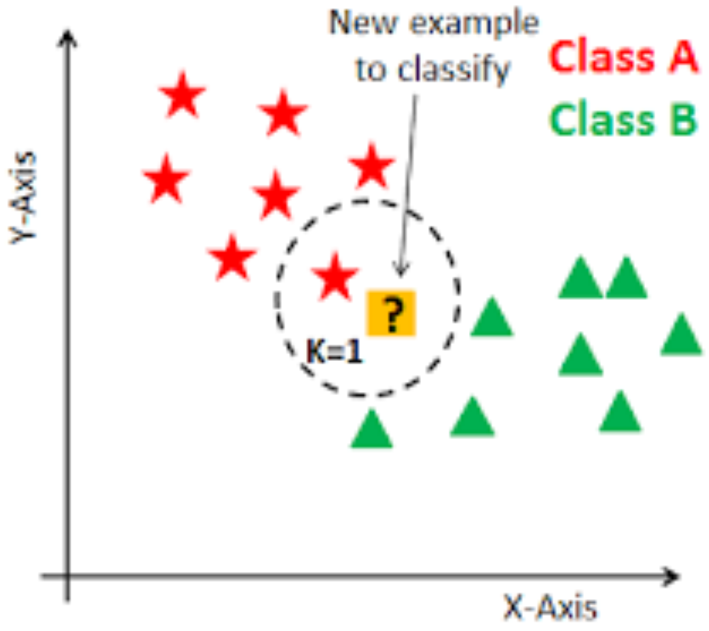
1.Speed

2.Sp. Def
3.HP
4.Defence
5.Sp. Atk

Whereas 2 features, Rock and Flying were found to have a close to 0 weight, meaning they play no role in deciding either of the class labels. For Pokemon being Not Legendary, the most important features are the categorical features such as Bug, Poison Normal, Fairy, Dark. In reality, there exists no Bug-type Pokemon in the data that is Legendary which exactly correlates with the weights that we have achieved.

### 2.1 K nearest neighbours approach

In this approach, we will try to predict the query point by looking at the K-nearest neighbours of the query point. Each neighbour gets a weight value inversely proportional to the distance from the query point. The final label is determined by the weighted sum of each class label corresponding to the K number of neighbours set that lie in the vicinity of the query point.[7]



$$weight = 1/distance$$

The distance metric is calculated with the euclidean measure which is defined as

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

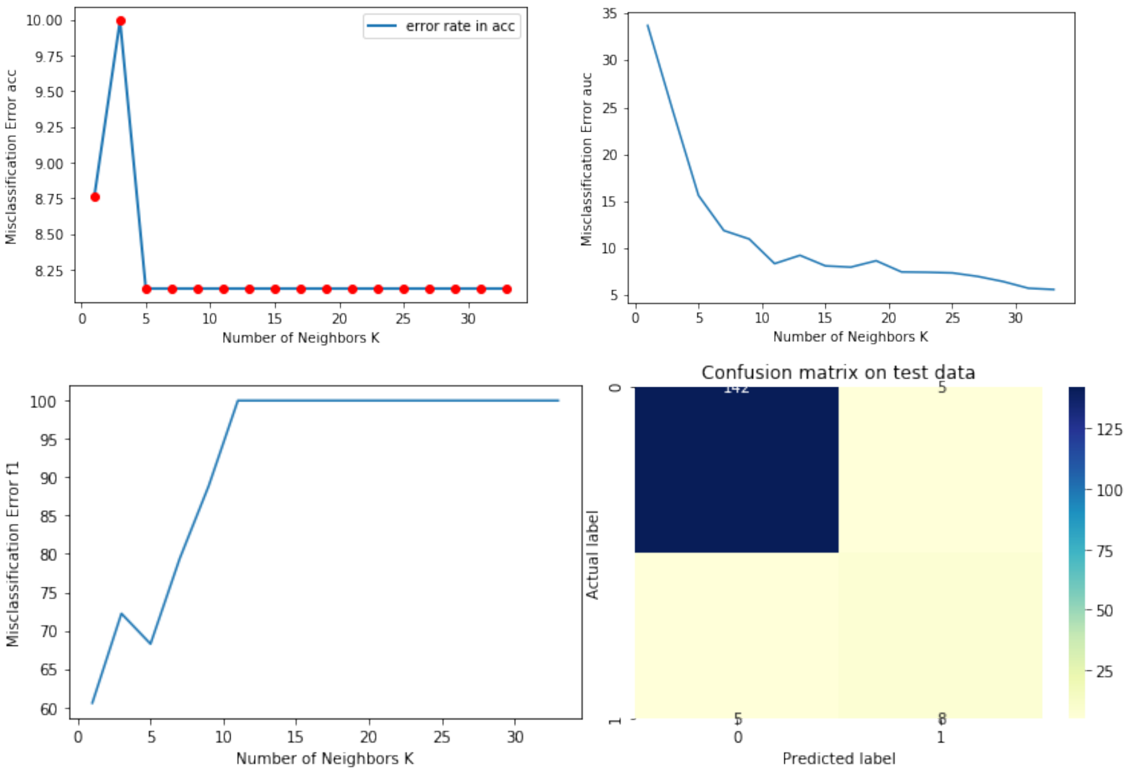
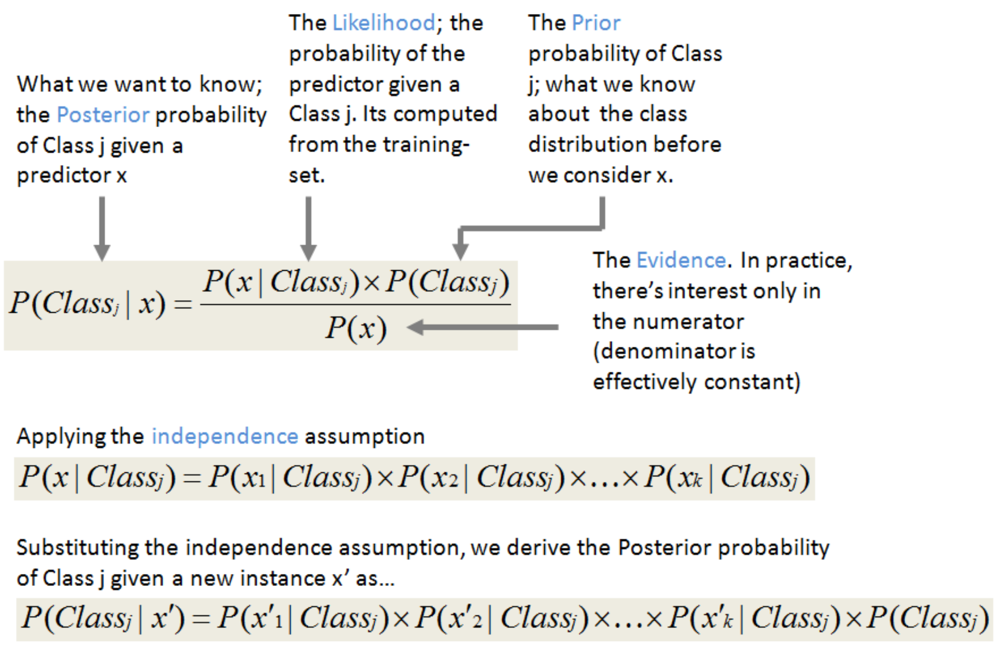


Fig2: (a) error in accuracy (b) error in abc curve (c) F-1 score loss (d)Confusion Matrix

## 2.2 *Probabilistic Naive Bayes approach*

In general Naive Bayes approach assumes conditional independence among the given features, meaning,

$$P(x_1, x_2 \dots, x_k \mid \text{Class}_j) = P(x_1 \mid \text{Class}_j) \times P(x_2 \mid \text{Class}_j) \times \dots \times P(x_k \mid \text{Class}_j)$$



What we want to know; the Posterior probability of Class j given a predictor x

The Likelihood; the probability of the predictor given a Class j. Its computed from the training-set.

The Prior probability of Class j; what we know about the class distribution before we consider x.

$$P(Class_j \mid x) = \frac{P(x \mid Class_j) \times P(Class_j)}{P(x)}$$

The Evidence. In practice, there's interest only in the numerator (denominator is effectively constant)

Applying the independence assumption

$$P(x \mid Class_j) = P(x_1 \mid Class_j) \times P(x_2 \mid Class_j) \times \dots \times P(x_k \mid Class_j)$$

Substituting the independence assumption, we derive the Posterior probability of Class j given a new instance x' as...

$$P(Class_j \mid x') = P(x'_1 \mid Class_j) \times P(x'_2 \mid Class_j) \times \dots \times P(x'_k \mid Class_j) \times P(Class_j)$$

Or in our example can be understood as,

$$P(x_1, x_2 \dots, x_k \mid \text{Legendary}) = P(x_1 \mid \text{Legendary}) \times P(x_2 \mid \text{Legendary}) \times \dots \times P(x_k \mid \text{Legendary})$$

or

$$P(x_1, x_2 \dots, x_k \mid \text{Legendary}) = \textstyle\prod_i P(x_i, \mid \text{Legendary})$$

We can thus interpret the Probability of a class given the datapoint(x') as, P(legendary|x’) = [P(x1|Legendary) × P(x2|Legendary) ×……….× P(x…n| legendary)]×P(Legendary)

If there is an occurrence of rare element say P(Bug | Legendary) which is 0 to prevent this feature to dominate we apply Laplace smoothing [8] parameter (**α**) defined as,

$$\hat{\theta}_i = \frac{x_i + \alpha}{N + \alpha d} \qquad (i = 1, \ldots, d),$$

To avoid the numerical underflow we will then apply a logarithmic function,

log(P(Legendary|x’))=log(P($x_1$ | Legendary)) + log(P($x_2$ | Legendary)) +…+ log(P($x_k$ | Legendary)) + log(P(Legendary))

Since this approach works on probability hence we need not have to standardise the data.

### 2.2.1 Complement Naive Bayes

This version of Naive Bayes is very well suited for Imbalanced datasets, the Laplace smoothing is defined as,

$$\hat{\theta}_{ci} = \frac{\alpha_i + \sum_{j:y_j \neq c} d_{ij}}{\alpha + \sum_{j:y_j \neq c} \sum_k d_{kj}}$$

$$w_{ci} = \log\hat{\theta}_{ci}$$

$$w_{ci} = \frac{w_{ci}}{\sum_j |w_{cj}|}$$

This approach looks at the presence of the individual feature in all the other classes(Pokemon not being legendary given a feature).

Upon training it was found the optimal smoothing parameter ($\alpha$) was found to be **0.001**.

### 2.2.2 Gaussian Naive Bayes

Since our data has both categorical features (transformed using one-hot encoding) and the numerical real-valued features. Hence we will try to map the Gaussian function and assume that the likelihood is Gaussian in nature[9].

A Gaussian function can be defined as,

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

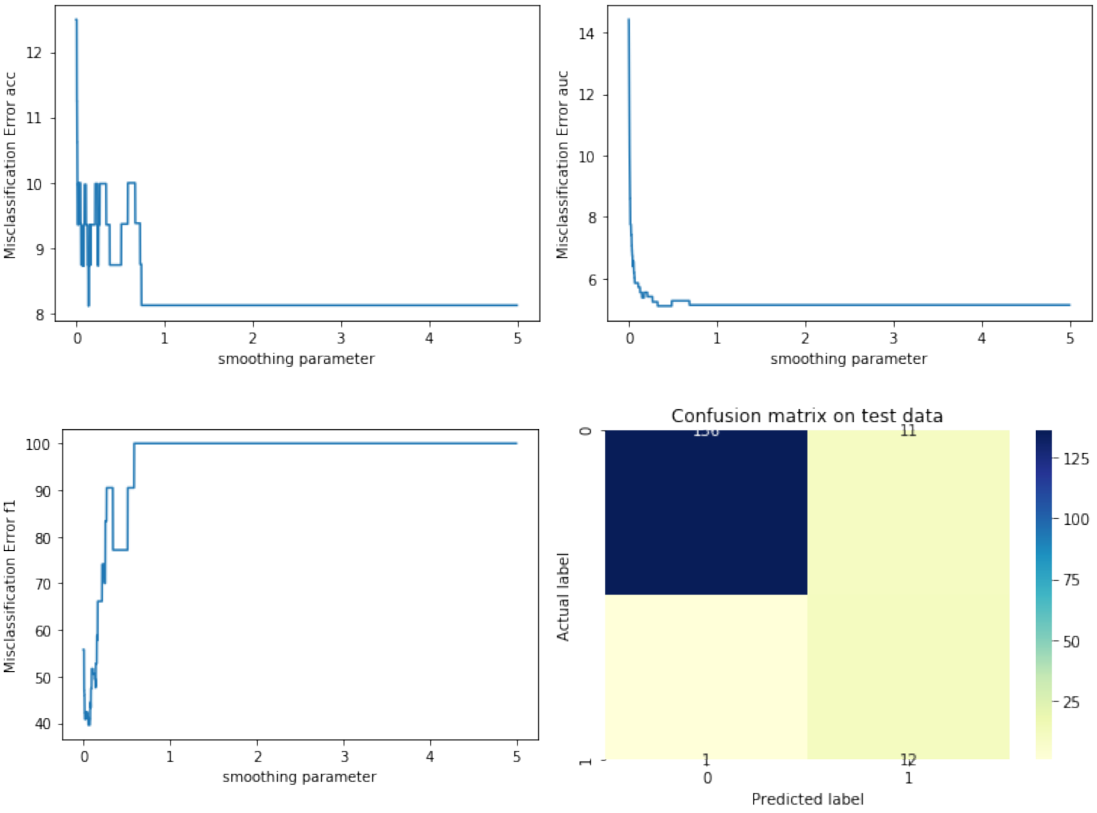


Fig2: (a) error in accuracy (b) error in abc curve (c) F-1 score loss (d)Confusion Matrix

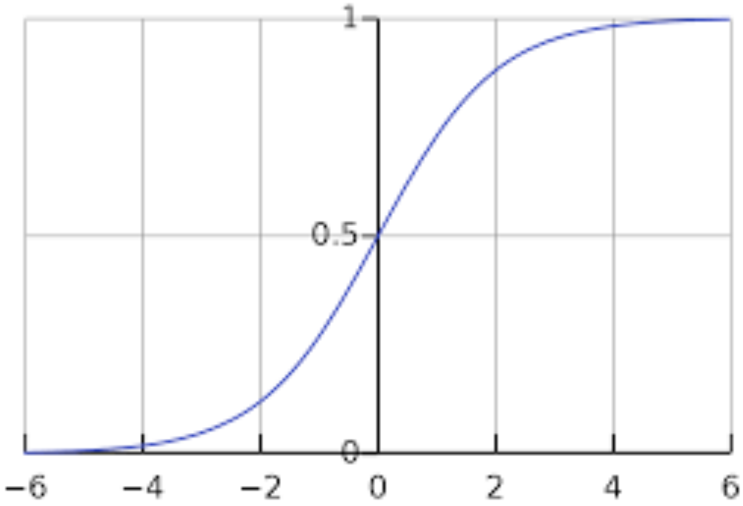
Upon training, it was found the optimal smoothing parameter (**α**) was found to be **0.035**.

## 2.3 Logistic Regression

In this approach we are trying to map a linear plane that can be fit into the n-dimensional space, we will use this plane to predict the class label of a point by using a squashing function known as a sigmoid function.

where the sigmoid function is defined as

$$P(y|\mathbf{x}) = \frac{1}{1 + e^{-y(\mathbf{w}^{\mathbf{T}}\mathbf{x}+b)}}.$$



Here is the equation of the hyperplane that separates both the classes, where the “w” is defined as the weight vector corresponding to each feature while “b” is defined as the intercept term.

Logistic regression is dependent upon the weight vectors hence we need to standardize the equation.

For any point that lies on the decision boundary of the hyperplane the probability value would be 0.5 as I will be 0
In addition to the sigmoid function, we also add a regularisation[10] term defined as

$$\lambda|W|^{n}|$$

Where λ is defined as the regularisation parameter to control the overfitting of the decision boundary plane.

so it tends to infinity we will have the decision boundary that quite easily separates the two classes. however, In order to avoid overfitting, the regularisation term controls the parameters since it will Increase drastically.
When n in the above equation is 1 it’s called l1 regularisation and this generates a sparse matrix as W consists of very small values, and it will turn all the features that are unimportant such as “Rock” and “Flying” to be 0.

The weights inferred from the logistic regression can be directly correlated with feature importance as higher is the weight the more important is the feature in predicting the positive class which is the anomalies (Legendary type pokemon).

Upon training it was found with upsampling the data, the optimal regularisation parameter (λ) was found to be 0.01 with the l1 regularisation.

## 2.4 Support vector Machines

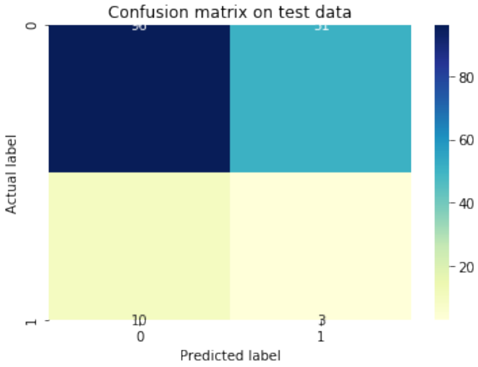
SVM’s are also known as large margin classifiers, They use the closest point from each class to draw their decision boundary and hence the points that fall on this line are known as support vectors.

Since the decision boundary obtained has to have an equal maximum distance from each of the support vectors. We will use a variety of kernels to draw and project a decision boundary in a high dimensional space.

In support vector machines[11] we try to achieve the class label by repeatedly changing the kernel function and the C value which is defined as,

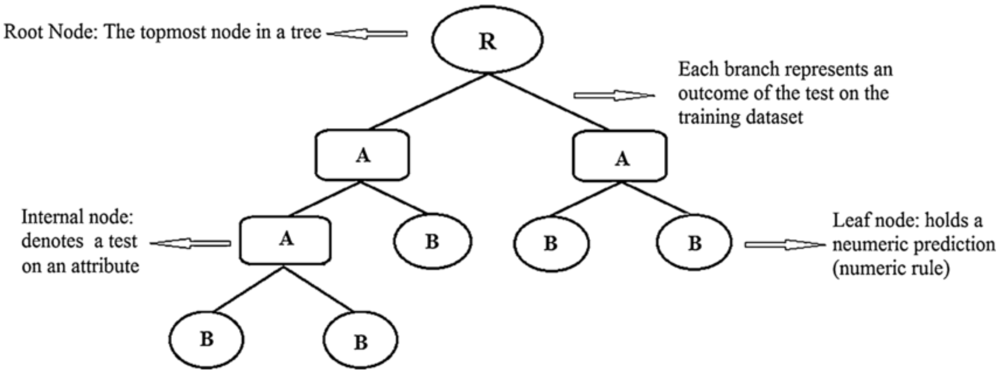
$$C = {}^{1}\!/_{\lambda}$$

Upon training it was found with upsampling the data, the optimal smoothing parameter **(C) was found to be 0.1 with the polynomial kernel of degree 3.**



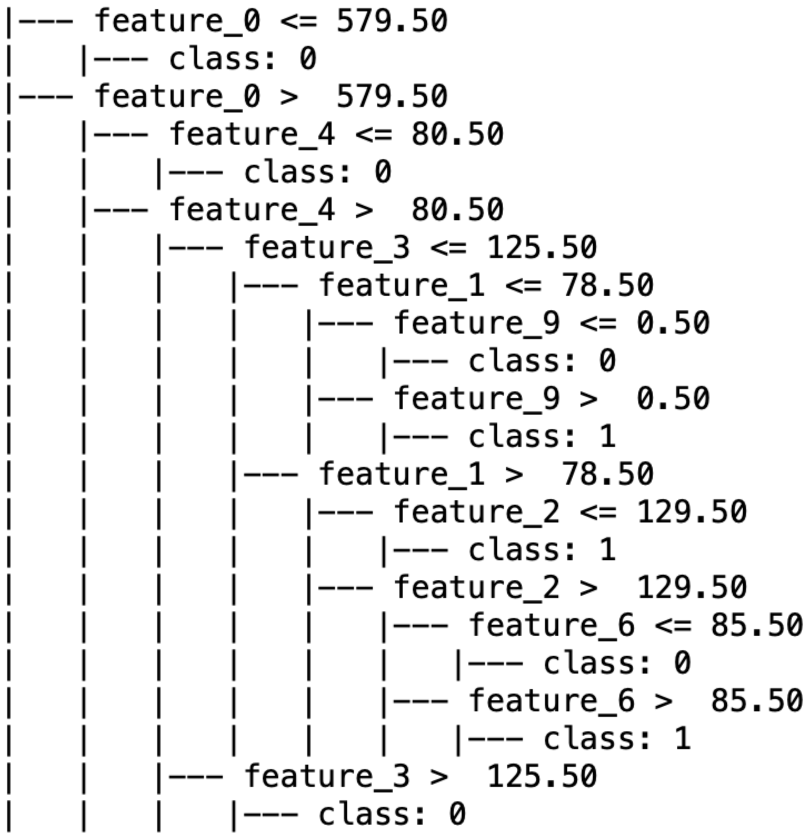
## 2.5 Decision trees

In this approach, we are trying to classify a data point to label using a tree-based approach by asking If-else questions based on the outcome of the tree.

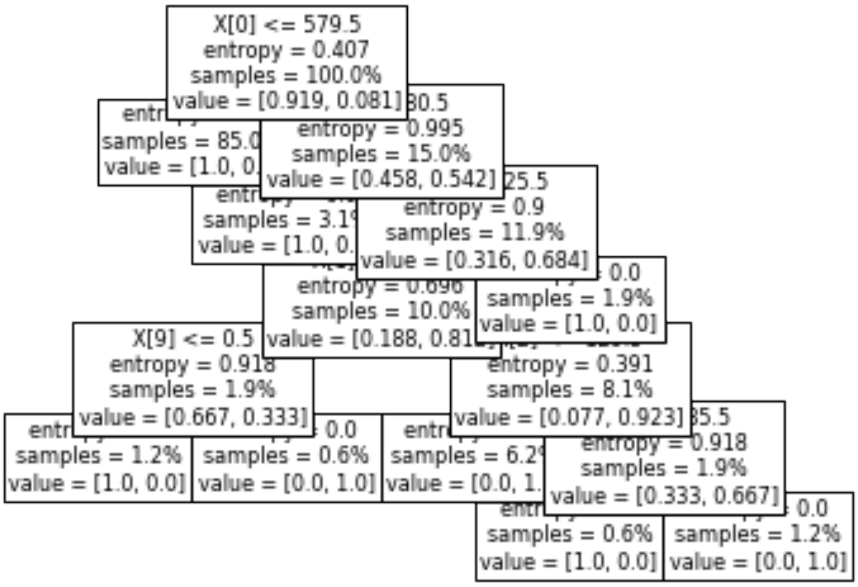

Root Node: The topmost node in a tree
R
Each branch represents an outcome of the test on the training dataset
A
A
Internal node: denotes a test on an attribute
A
B
B
B
Leaf node: holds a neumeric prediction (numeric rule)
B
B

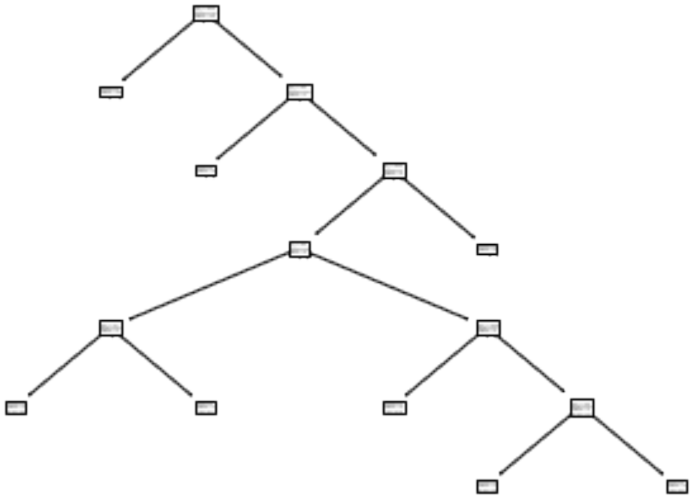
The decision trees obtained are very easy to interpret and have low run time for evaluation, however, there is a severe drawback with this approach which is the overfitting as the tree depth increases, meaning as soon as we reach the deepest node we could very easily overfit on our model.

The decision tree obtained by fitting in test data is:



```
|--- feature_0 <= 579.50
|   |--- class: 0
|--- feature_0 >  579.50
|   |--- feature_4 <= 80.50
|   |   |--- class: 0
|   |--- feature_4 >  80.50
|   |   |--- feature_3 <= 125.50
|   |   |   |--- feature_1 <= 78.50
|   |   |   |   |--- feature_9 <= 0.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_9 >  0.50
|   |   |   |   |   |--- class: 1
|   |   |   |--- feature_1 >  78.50
|   |   |   |   |--- feature_2 <= 129.50
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_2 >  129.50
|   |   |   |   |   |--- feature_6 <= 85.50
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- feature_6 >  85.50
|   |   |   |   |   |   |--- class: 1
|   |   |--- feature_3 >  125.50
|   |   |   |--- class: 0
```


X[0] <= 579.5
entropy = 0.407
samples = 100.0%
value = [0.919, 0.081]
samples = 85.0
value = [1.0, 0.
entropy = 0.995
samples = 15.0%
value = [0.458, 0.542]
samples = 3.1%
value = [1.0, 0.
entropy = 0.9
samples = 11.9%
value = [0.316, 0.684]
samples = 1.9%
value = [1.0, 0.0]
entropy = 0.696
samples = 10.0%
value = [0.188, 0.81
X[9] <= 0.5
entropy = 0.918
samples = 1.9%
value = [0.667, 0.333]
entropy = 0.391
samples = 8.1%
value = [0.077, 0.923]
samples = 1.2%
value = [1.0, 0.0]
samples = 0.6%
value = [0.0, 1.0]
samples = 6.2%
value = [0.0, 1.
entropy = 0.918
samples = 1.9%
value = [0.333, 0.667]
samples = 0.6%
value = [1.0, 0.0]
samples = 1.2%
value = [0.0, 1.0]



## 2.6 Random forest

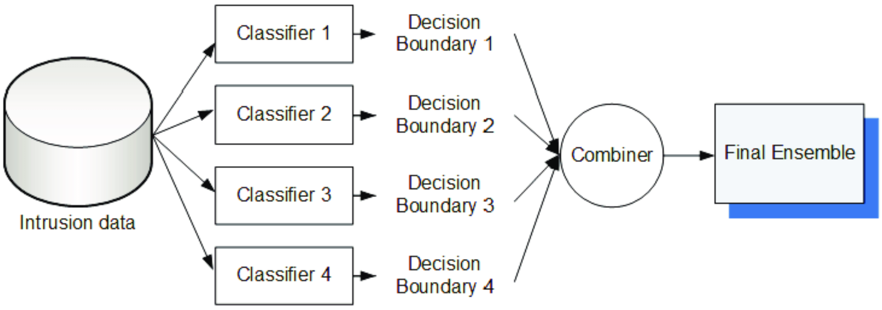
Random Forest is an approach where more than one relatively not correlated models (trees) operating as a single model that can easily outperform any of the individual tree-based models. The main idea is to not correlate each of these trees as decisions trees are very sensitive to the data they are trained on, small changes to the training set can result in an entirely different tree structure. Random forest exploits this fact by letting each individual tree to randomly sample from the dataset with replacement and thus resulting in a different tree structure in every seed. This process is known as bagging.

- The question asked about the data is based on the value of a feature. Each question has a binary end it can be either a True or False answer that divides the node. Based on the answer to the above-asked question, a data point escalates to the bottom of the tree.
- Samples: The total number of observations made for that node.
- Value: The number of samples in each label.
- Class: The majority classification for points in the node. In the case of leaf nodes, this is the prediction for all samples in the node.

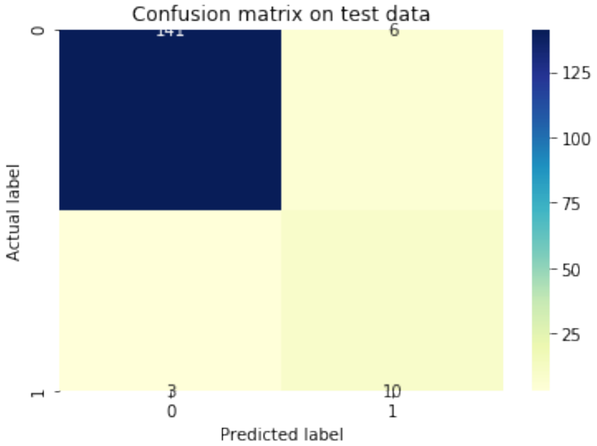
## 2.7 Ensemble voting classifier

The classifier works on combining each of the above-mentioned algorithms into a combined model that predicts the sample based on a probabilistic soft vote where each of the above-mentioned algorithms acts as a weak learner. The idea is to put many of these weak learners together in a model that would make a really strong classifier.

The diagram below explains the working of a voting classifier[12].



To get the maximum F1 score we will take a hard vote where the majority class label from each of the individual classifiers (weak learner) decides the outcome of the datapoint.



## 3 Methodology

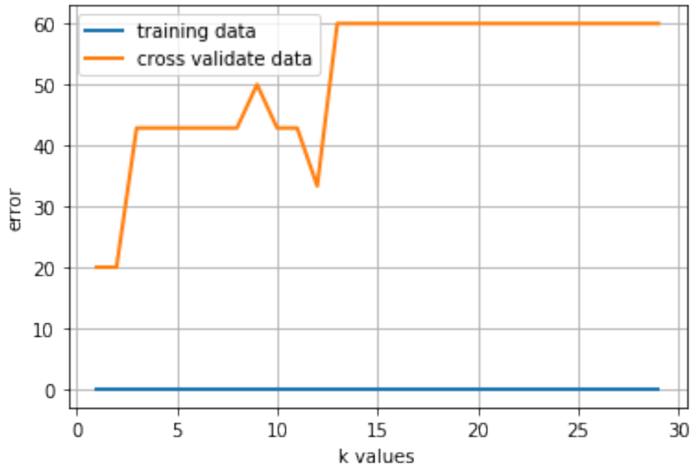
We have split the data into three parts:

1.**Train data:** consists of 70% of the randomly shuffled datapoint. This is the data that will be provided to the classifier to learn and hence our classifiers will try to mark the decision boundary based only on this data.

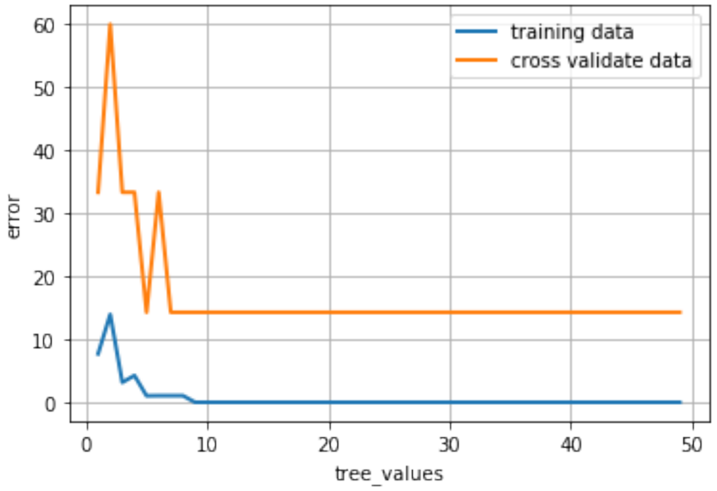
2.**Cross-validation:** consists of roughly 10% of the randomly shuffled datapoint. The main objective of the split is to tune the hyperparameters of the classifier that best performs in this data

**3.Test Data:** consists of roughly 20% of the randomly shuffled datapoint which will only be used to evaluate the data. The optimized hyperparameters from the cross-validation fold will be used to evaluate the performance on this fold.

The below graph explains hyperparameter tuning (K) was tuned on a cross-validation dataset to plot the F1 score.



Another graph that explains the hyperparameter to select the number of decision trees based on the F1 score achieved on the cross-validation data.



| List of algorithms | Single run F1 score | 3 fold CV F1 score | 3 fold CV ROC/AUC | 3 fold CV Accuracy | Optimal Hyperparameters |
|---|---|---|---|---|---|
| K nearest neighbour | 61% | 31.74% | 91.88% | 84.37% | K=5 |
| Compliment Naive Bayes | 30% | 16.73% | 62.48% | 74.34% | α=0.23 |
| Gaussian Naive Bayes | 66.66% | 57.65% | 92.92% | 90% | α=0.035 |
| Logistic Regression | 57.77% | 33.93% | 88.7% | 68.11% | λ = 4.19, L1 |
| Support Vector Machines | 64.70% | 45.95% | 92.34% | 89.40% | C=0.5<br>kernel=“poly” |
| Decision trees | 59.25% | 59.52% | 80.95% | 93.12% | “entropy” |
| Random forest | 64% | 54.52% | 92.44% | 93.12% | tree=5 |
| Ensemble Voting Classifier | 66.66% | 39.52% | 91.56% | 91.26% | voting=“Soft” |

## 4 Evaluation

Accuracy is not a good measure for imbalanced datasets, hence we are using F1 Score as the primary measure to compare and evaluate the performance of each model and also to tune the hyperparameter on the cross-validation dataset.

F1 score is defined as,

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

We have chosen this metric because it takes a balanced value of both precision and recall into account.

Along with the F1 score, we are also looking at the ROC/AUC score to measure the sensitivity of the model and the simple accuracy measure. The area under the ROC curve gives us a fairly clear picture of the model's performance.

## 5 Conclusion

The Ensemble classifier was the best approach to detect the outliers as it has a decent score in both the single and 3 fold run cross-validation on test data. Moreover, the prediction is based on not just one but many individual weak learners. Moreover, the maximum contribution to the Ensemble model is drawn from the Gaussian Naive Bayes model as it assumes that my features follow a gaussian Distribution which is True for most real-valued features such as HP and Attack strength of the Pokemons.

The result received is fairly limited to the size of the dataset, In most of the cases, prediction accuracy is very much dependent on row samples in the data. The more samples we have the better is the prediction. However, further splitting the dataset into cross-validation having roughly 10% of the total data will have very few points to map them as anomalies.

## References

1. https://www.kaggle.com/abcsds/pokemon.
2. Evaluation of convolutionary neural networks modeling of DNA sequences using ordinal versus one-hot encoding method In: Allen Chieng Hoon Choong
   Faculty of Cognitive Sciences and Human Development, Universiti Malaysia Sarawak, Kota
   Samarahan, Malaysia.
3. **SMOTE**: synthetic minority over-sampling technique,NV Chawla,
4. SvanteWold  KimEsbensen  PaulGeladi : Principal component analysis
5. Laurens van der Maaten, Geoffrey Hinton; 9(Nov):2579--2605, 2008: Visualizing Data using t-SNE
6. Akinori Fujino, Hideki Isozaki, Jun Suzuki: IMulti-label Text Categorization with Model Combination based on F1-score Maximization
7. Sahibsingh A. Dudani: The Distance-Weighted k-Nearest-Neighbor Rule
8. Laplacian smoothing and Delaunay triangulations: David A. Field
9. Estimating continuous distributions in Bayesian classifiers: George H.John
10. Feature selection, L1 vs. L2 regularization, and rotational invariance
11. Least Squares Support Vector Machine Classifiers: J.A.K. SuykensJ. Vandewalle

12. A weighted voting framework for classifiers ensembles: Ludmila I. Kuncheva