Mathematics in Machine Learning

Politecnico di Torino

Syed Ali Abbas
Student id: s292423
s292423@studenti.polito.it

Fall Session 2022

Abstract

The input is a set of Monte Carlo data generated for an imaging gamma-ray Cherenkov telescope and roughly triggered and pre-processed. These data fall into two categories: those arising from incident gamma rays and those resulting from hadronic showers. The data is a good testing ground for classification systems because there is only a weak distinction between signal (gamma) and background (hadrons).

Our research focuses on determining if a signal captured by a Cherenkov telescope is a gamma signal or a hadron signal (background). Simple classification was not a possibility in this case because, as you will see later, the data is not distributed evenly and is biased towards the gamma signal. So, in order to address this issue, we performed some exploratory data analysis (EDA), attempted to reduce bias using SMOTE, and then employed various classifiers to achieve the best possible results.

Contents

1 Introduction **3**

2 Data Analysis **5**
2.1 Box Plot 7
2.1.1 Explanation Box plot 7
2.2 Kernel Density Estimation 8
2.2.1 Explanation for KDE plots 9
2.3 Pair Plot 9
2.4 HeatMap 11

3 Pre Processing **12**
3.1 Detecting anomalies 12
3.1.1 Isolation Forest 12
3.2 Principal Component Analysis 14
3.2.1 Covariance 15
3.2.2 Eigenvectors and Eigenvalues 15
3.2.3 Dimensionality Reduction 16
3.3 Cross Validation 16
3.3.1 K-Fold 17
3.3.2 Sampling Bias 17
3.3.3 Stratified Sampling 17
3.4 Over sampling vs Under sampling 18
3.4.1 SMOTE 18

4 Model Selection **20**
4.1 Decision Tree 20
4.1.1 Advantages 20
4.1.2 Disadvantages 20

4.1.3 Splitting Criteria 21
4.2 Random Forest 21
4.2.1 Bootstrap aggregation 22
4.2.2 Hyper parameters 22
4.3 Logistic Regression 23
4.3.1 Decision Boundaries 24
4.4 Support Vector Machine(SVM) 24
4.5 K-Nearest Neighbour 27
4.5.1 Selection of K in KNN 27
4.5.2 Determining a point as neighbor 28
4.5.3 Working 29

5 Evaluation metrics **31**
5.1 Some Notations 31
5.2 Accuracy 32
5.3 Recall 32
5.4 Precision 32
5.5 F1-score 32

6 Model Evaluation **33**

Chapter 1

Introduction

The MAGIC telescopes are one of three largest IACTs (Imaging Atmospheric Cherenkov Telescopes) currently in operation for observing gamma rays in the TeV regime. Since its inception in 2003, MAGIC has published data from over 60 different sources, the majority of which are blazars. From published data, MAGIC already supplies astronomical .fits files containing fundamental final scientific products including spectral energy distributions, light curves, and skymaps. With a mirror surface of 236m2, the MAGIC telescope is the world's largest and most advanced Cherenkov telescope, as well as the ground device with the lowest threshold and maximum sensitivity in the cosmic-ray energy domain below around 200 GeV.

The two MAGIC telescopes, each with a reflector diameter of 17 meters, are the world's most sensitive Cherenkov telescopes, notably in the energy range below 200 gigaelectronvolts (GeV). Their line of sight is focused on gamma ray emitters with energies ranging from 30 GeV to 100 TeV. (tera-electronvolts). As a result, MAGIC may cover a wide range of energy. The twin telescopes are located 2,200 meters above sea level on the Canary Island of La Palma, where bright sky and minimal light pollution provide ideal watching conditions. The multinational partnership comprising roughly 165 astrophysicists from 24 research institutes in eleven countries is led by the Max Planck Institute for Physics (MPP). They are in charge of the telescopes' construction, operation, and maintenance as a team. Astrophysicists can use MAGIC to get first-class data for scientific research into enigmatic objects and the universe's most violent processes.

Since 2003 and 2009, respectively, the MAGIC telescopes have been in operation. The mechanical framework, imaging cameras, and calibration

system were all developed and built in part by the MPP.

MAGIC has yielded a slew of important scientific breakthroughs since its inception. The number of h occurrences is undercounted for technical reasons. The h class reflects the majority of events in real data.

Chapter 2

Data Analysis

The input is a portion of Monte Carlo data that was approximately triggered and pre-processed for an imaging gamma-ray Cherenkov telescope. Data comes from one of two classes, incoming gamma rays or hadronic showers. Dataset has 11 coloumns and 19,020 instances

- **fLength**: major axis of ellipse [mm]
- **fWidth**: minor axis of ellipse [mm]
- **fSize** : 10-log of sum of content of all pixels [in phot]
- **fConc**: ratio of sum of two highest pixels over fSize [ratio]
- **fConc1**: ratio of highest pixel over fSize [ratio]
- **fAsym**: distance from highest pixel to center, projected onto major axis [mm]
- **fM3Long**: 3rd root of third moment along major axis [mm]
- **fM3Trans**: 3rd root of third moment along minor axis [mm]
- **fAlpha** : 3rd root of third moment along minor axis [mm]
- **fDist** : distance from origin to center of ellipse [mm]
- **class** : g,h: gamma (signal), hadron (background)

All of the features are numerical except the one we have to classify. During preliminary analysis no missing values were found.

	fLength	fWidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	fDist
count	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000
mean	53.250154	22.180966	2.825017	0.380327	0.214657	-4.331745	10.545545	0.249726	27.645707	193.818026
std	42.364855	18.346056	0.472599	0.182813	0.110511	59.206062	51.000118	20.827439	26.103621	74.731787
min	4.283500	0.000000	1.941300	0.013100	0.000300	-457.916100	-331.780000	-205.894700	0.000000	1.282600
25%	24.336000	11.863800	2.477100	0.235800	0.128475	-20.586550	-12.842775	-10.849375	5.547925	142.492250
50%	37.147700	17.139900	2.739600	0.354150	0.196500	4.013050	15.314100	0.666200	17.679500	191.851450
75%	70.122175	24.739475	3.101600	0.503700	0.285225	24.063700	35.837800	10.946425	45.883550	240.563825
max	334.177000	256.382000	5.323300	0.893000	0.675200	575.240700	238.321000	179.851000	90.000000	495.561000

Figure 2.1: Statistics of Numerical Features

The target class distribution is shown in the figure below.

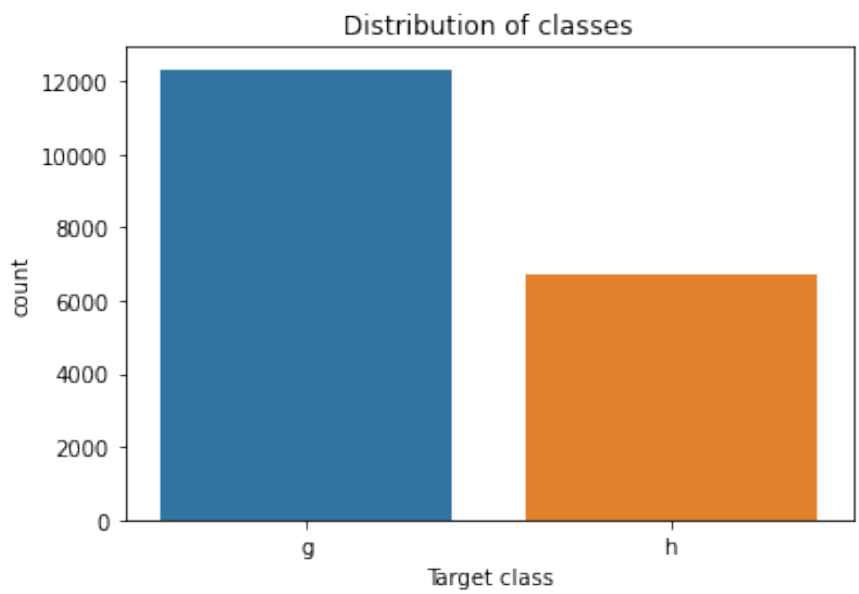


Figure 2.2: Target Class Distribution

We can see that the gamma signal class has about double the count of the hadron (background) signal class. It's also common knowledge that most algorithms are sensitive to unbalanced data, and that the algorithms' output

will be biased towards the majority class. Later in the preprocessing phase, we'll see how we can solve this problem.

2.1 Box Plot

A boxplot is a standardized method of displaying data distributions using a five-number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum"). It can provide information about outliers and their associated values. It can also determine whether the data is symmetrical, how tightly the data is packed, and whether or not the data is skewed. If we present a boxplot picture, it will be clearer.

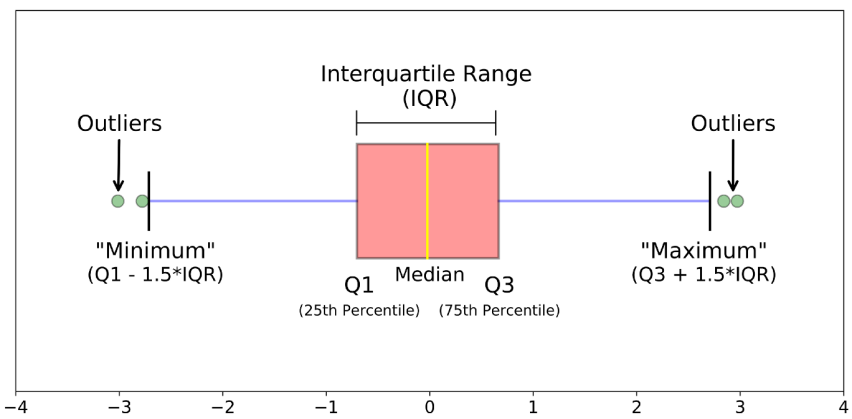


Figure 2.3: Box Plot

To extract important insights, the distribution of all features was shown in figure 2.4 using boxplots..

2.1.1 Explanation Box plot

We will need to address anomaly identification and removal during the preprocessing phase, as the graphs suggest that practically every feature has outliers..
Another discovery is that when features like fAlpha, fLength, and fWidth

are plotted independently for each plot, they exhibit little variance in their distribution. This observation offers us an understanding of feature value at a very early stage (i.e Data exploration).

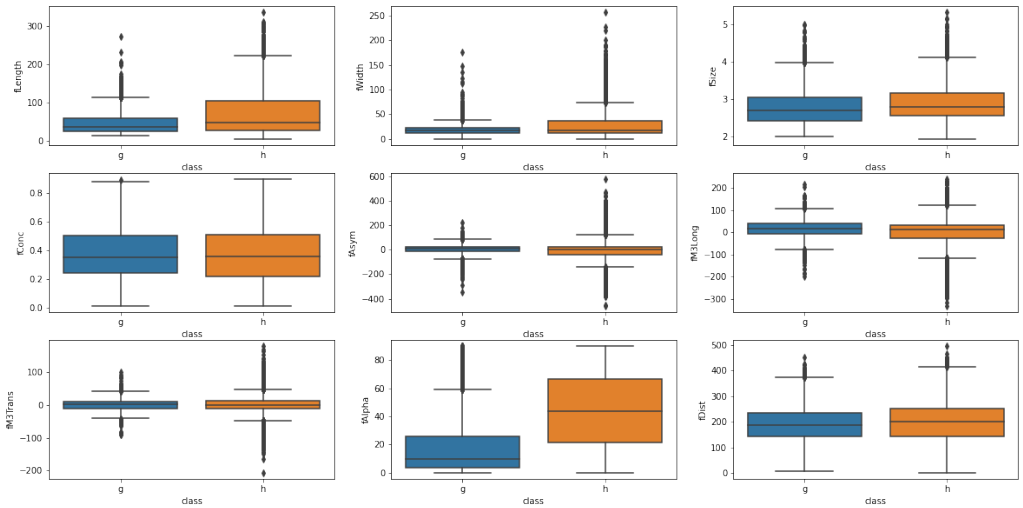


Figure 2.4: BoxPlots

2.2 Kernel Density Estimation

Kernel density estimation, often known as KDE, is a non-parametric approach that allows you to construct a smooth curve from a set of data. To get a destiny estimate, KDE centers a kernel function at each data point and smooths it. KDE was created because histograms are not smooth and are dependent on the width of the bins and the endpoints of the bins. KDEs solve this problem by offering smoother curves. This can be handy if you only want to see the "shape" of data, as a continuous alternative to the discrete histogram. Kernel Density Estimation smooths data by convolving each point with a 'kernel.' Each point is represented as the center for a kernel density function and the final curve is normalized convolution(sum) of all kernels at that point. It helps to visually observe that if the distribution of all the features are separate kernel density functions or not if yes, we might want to apply LDA on the data otherwise think of something else. The Kernel Density Estimation plots for the data have been in figure 2.5 :

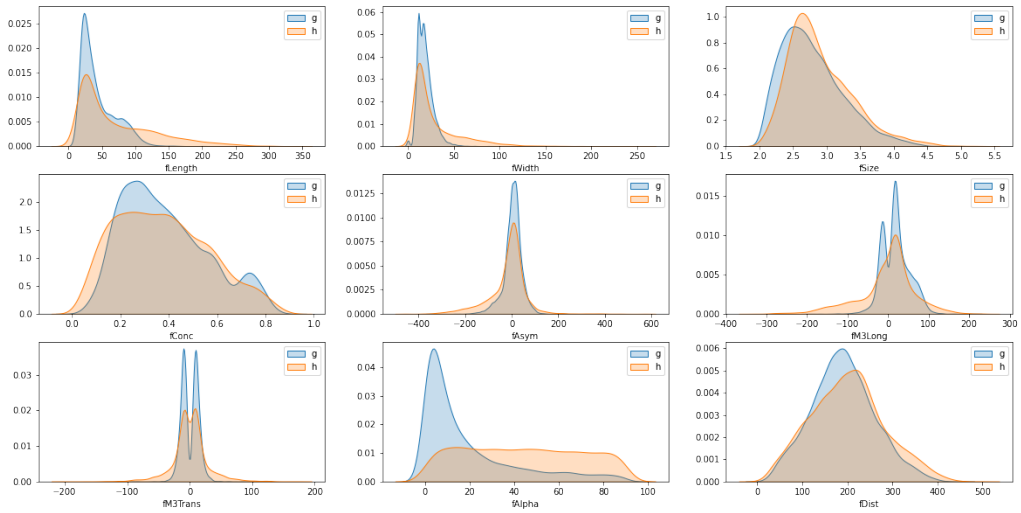


Figure 2.5: Feature Distributions

2.2.1 Explanation for KDE plots

The KDE plots demonstrate that the feature distributions are overlapping across classes. This type of distribution does not comply with algorithms such as Linear discriminant analysis where we presume that the inter class distributions are properly separated. In the section on dimensionality reduction, we'll go over this in greater detail.

2.3 Pair Plot

Pair plots are a quick and easy way to see how each variable affects the others. It generates a matrix of relationships between each variable in the data for quick data analysis. We can observe both the distribution of a single variable and the relationship between two variables using a pair plot. Pair plots are a wonderful way to spot tendencies for further investigation, and they're simple to use in Python. The diagonal histogram shows the distribution of a single variable, whereas the upper and lower triangular scatter plots demonstrate the relationship (or lack thereof) between two variables. The pairplot for our data is shown below.

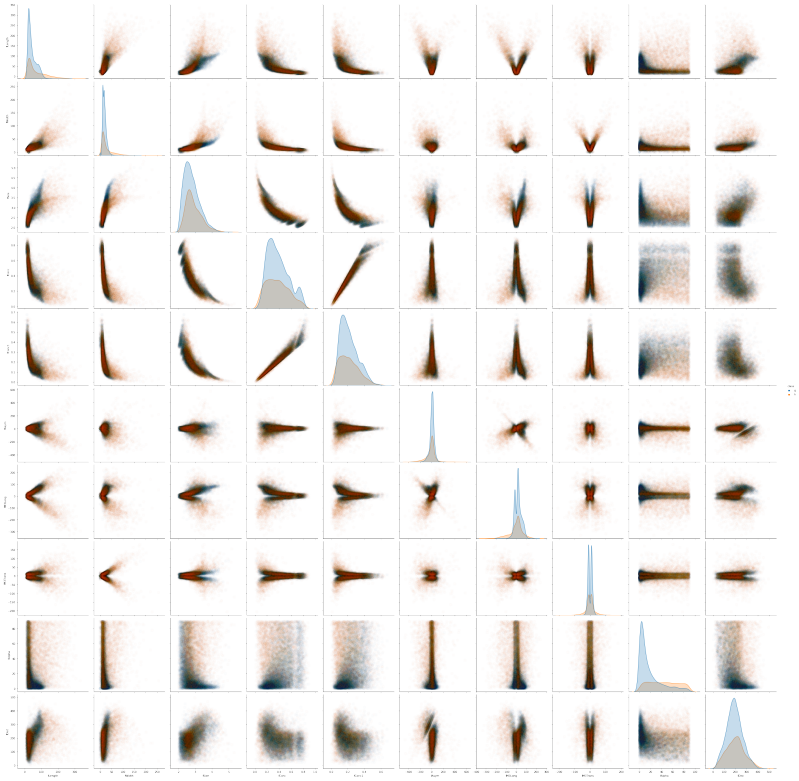


Figure 2.6: PairPlot

Because density charts on the diagonal make it easier to compare distributions between classes than stacked bars, we employed Kernel Density Estimation distribution plots at diagonals. Because there is a lot of overlap (known as overplotting) on these images, changing the transparency of the scatter plots improves readability.

Pairs plots are a useful tool for quickly exploring dataset distributions and relationships. The Pair Grid class in Seaborn provides a simple default way for creating pair plots that may be tweaked and enhanced. In a data analysis project, a large percentage of the value is often found in the simple display of data, rather than the showy machine learning. A pair plot is a unique tool in the data analysis pipeline that gives us a complete first look at our data..

2.4 HeatMap

A heat map chart, also known as a heatmap, is a two-dimensional visual representation of data in which values are encoded in colors to provide a quick and insightful perspective of data. Figure 2.7 shows a heat map for the dataset. With a correlation of 0.98, the map shows that fconc and fconc1 are practically overlapping features. As a result, we can drop one of them without sacrificing in- formation. Furthermore, because the first three features have a correlation of greater than 0.7, we may want to map these features onto another dimension for a more compact representation (s). More about this under the PCA section.

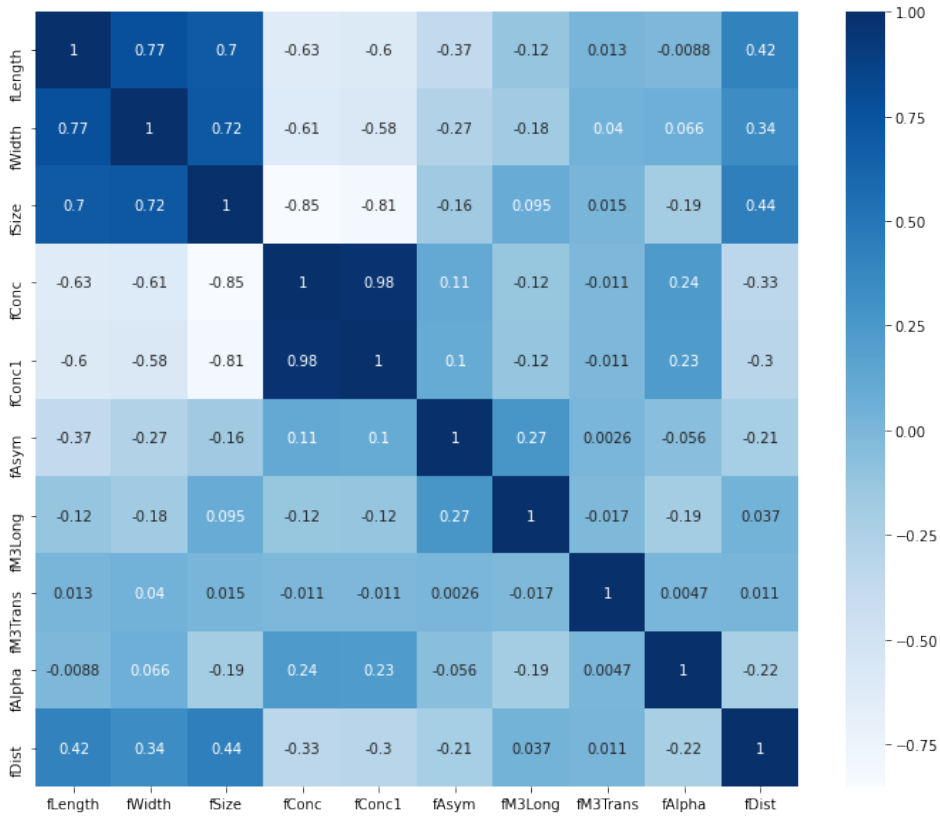


Figure 2.7: HeatMap

Chapter 3

Pre Processing

3.1 Detecting anomalies

3.1.1 Isolation Forest

The isolation forest algorithm is a machine learning approach for detecting anomalies. It's a Decision Tree Classifier-based unsupervised learning algorithm. It isolates the outliers by randomly selecting a feature from the given collection of features and then randomly selecting a split value between the max and min values of that feature. Outliers are less common than normal observations and differ in terms of values (they lie further away from the regular observations in the feature space).As a result, they should be detected closer to the tree's root (shorter average route length, i.e., the number of edges an observation must transit in the tree moving from the root to the terminal node) with fewer splits.

Figure 3.1 from [1] illustrates the concept of identifying a normal vs. abnormal observation. A normal point (on the left) necessitates the identification of more partitions than an atypical point (right).

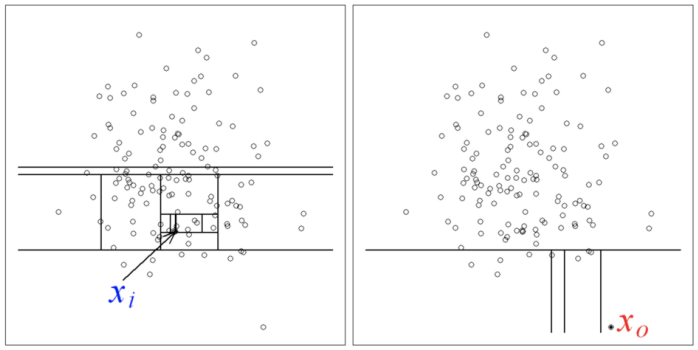


Figure 3.1: Normal vs Abnormal observations

Anomaly scores, like other outlier identification approaches, are necessary for decision-making. It is defined as follows in the case of Isolation Forest:

$$s(x, n) = 2^{\frac{-E(h(x))}{c(n)}}$$

where h(x) represents the path length of observation x, c(n) represents the average path length of unsuccessful searches in a Binary Search Tree, and n represents the number of external nodes

Each observation is assigned an anomaly score, which can be used to make the following decision:

- A score near 1 denotes anomalies.
- A score of less than 0.5 implies that the observations are normal.
- If all of the scores are close to 0.5, the overall sample does not appear to have any evident anomalies.

We used isolation forest to delete outliers, and we were able to remove 1902 outliers from our data set, leaving us with 17118 instances, as isolation forest removes the entire row from the dataset.

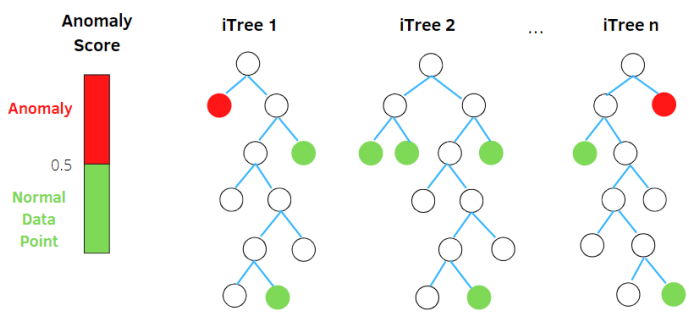


Figure 3.2: Isolation Forest

3.2 Principal Component Analysis

We evaluate complicated data, i.e. multi-dimensional data, in real-world data analysis jobs. We plot the data to see if there are any patterns, or we use it to train machine learning models. Consider a data point x as a physical thing; dimensions are simply a point of view, such as where the data is located when viewed from the horizontal or vertical axes. As data becomes more complex, it becomes more difficult to perceive it and conduct calculations on it. So, how do you lower the size of a data-set

- Keep only the most crucial dimensions.

PCA creates a new set of dimensions (or foundation of views) that are all orthogonal (and so linearly independent) and prioritized according to the variance of data along them. It means that the principleaxes that are more important come first. (More variance/spread out data = more important).

How does PCA work -

- Compute the vector μ containing mean of all the features in the dataset.
- Compute the centered data Matrix: $B = X - \mu$
- Calculate the covariance matrix of data: $S = \frac{1}{n}BB^T$
  Where S has a property of being symmetric and hence can be diagonalized. $S = PDP^T$

- Calculate the eigenvalues and eigenvectors over covariance matrix. Where we require that: $\lambda_1 \geq \lambda_2 ..... \geq \lambda_n \geq 0$
- Choose the principle components based on the explained variance threshold (e.g. 0.9) or the number of components to keep.

3.2.1 Covariance

It is a measure of the extent to which corresponding elements from two sets of ordered data move in the same direction.

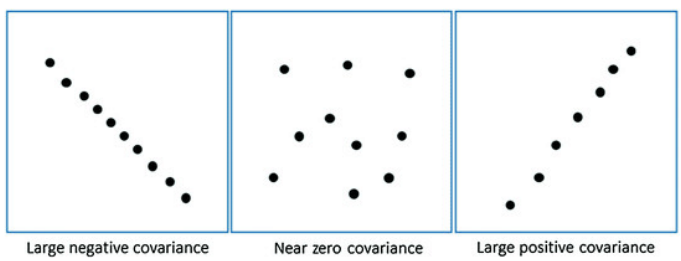


Figure 3.3: Covariance

Positive covariance indicates that X and Y are positively connected, meaning that as X rises, so does Y. Negative covariance represents the inverse relationship. Zero covariance, on the other hand, indicates that X and Y are unrelated.

3.2.2 Eigenvectors and Eigenvalues

The "core" of a PCA is represented by the eigenvectors and eigenvalues of a covariance (or correlation) matrix: The directions of the new feature space are determined by the eigenvectors (principal components), and their magnitude is determined by the eigenvalues. In other words, the data variance along the new feature axes is explained by the eigenvalues.

3.2.3 Dimensionality Reduction

PCA is an unsupervised dimensionality reduction technique that aims to capture the majority of a datset's variation. Linear discriminant analysis, which is based on supervised learning, is another intriguing technique. LDA, on the other hand, Assumes that the class wise distribution for each feature is well segregated, which is not the case in our dataset. We only have a few features, but the heatmap indicates a strong relationship between fsize, fwidth, and flength, thus we expect PCA to give us a more compact representation of the data. Figure 3.4 depicts explained variation and cumulative variation.

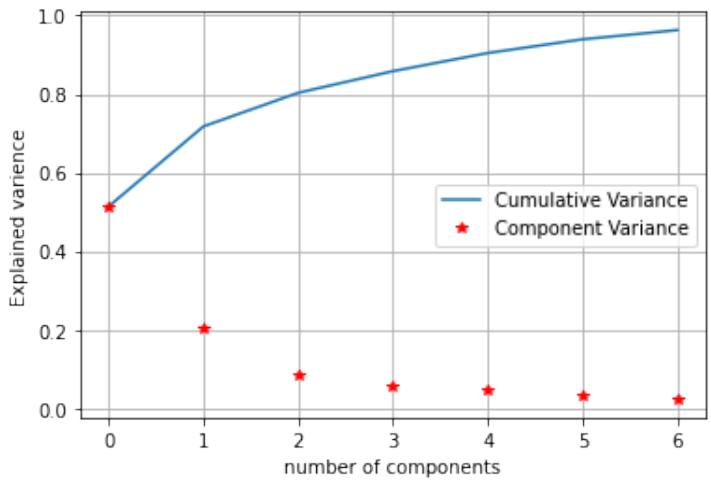


Figure 3.4: Principle Component Analysis

3.3 Cross Validation

Cross validation is a method for determining how well a statistical analysis generalizes to a different collection of data. It's a method of evaluating machine learning models that involves training numerous models on subsets of the available input data and then evaluating them on the complementary subset. With cross-validation, we have a good probability of detecting overfitting quickly.

3.3.1 K-Fold

The complete training data set is initially divided into k equal sections. The hold out (testing) set is preserved, while the remaining k-1 components are used to train the model. The trained model is then put to the test on the holdout data. The above procedure is repeated k times, with the holdout set changing each time. As a result, every data point has an equal chance of being included in the test set. Typically, k equals 3 or 5. It can be expanded to much higher values, such as 10 or 15, but this is incredibly computationally expensive and time-consuming.

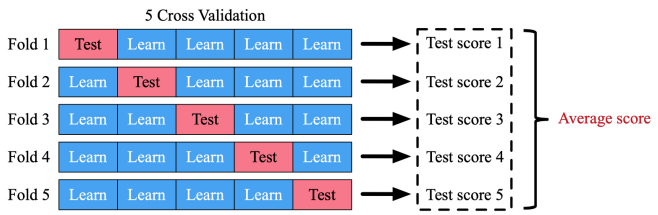


Figure 3.5: Crossvalidation with k=5

3.3.2 Sampling Bias

Because of non-random factors, the samples of a stochastic variable that are collected to determine its distribution are erroneously picked and do not represent the true distribution. Because specific values of the variable are systematically under-represented or over-represented in comparison to the true distribution of the variable, sampling bias occurs frequently.

3.3.3 Stratified Sampling

To eliminate sampling bias, stratified sampling must be used, which implies that the sample from each class must be of the same proportion in each split. Simple random sampling, on the other hand, entails selecting data from the entire population at random so that each conceivable sample has an equal chance of occurring. Stratified sampling guarantees that each subset of the population is adequately represented in the sample. As a result, stratified random sampling ensures that the entire population is covered.

3.4 Over sampling vs Under sampling

We frequently encounter datasets with unbalanced class distributions. As a result, the classifiers are biased in favor of the majority class. Undersampling the dominant class or oversampling the minority class are two options for dealing with this dilemma. Under smapling, the number of dataset instances will be reduced, which is not advised. On the other hand, the minority class may be exaggerated. There are various options for doing so. SMOTE is an extremely popular technique (Synthetic Minority Oversampling Technique).

Figure 3.6: Undersampling vs Oversampling

3.4.1 SMOTE

Using this method, new data points for the minority class are created by combining existing data points. This strategy will be used to train all of the classifiers. The following diagram depicts how this strategy works:

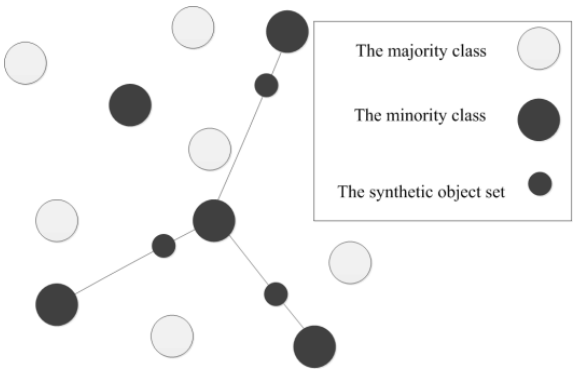


Figure 3.7: SMOTE

Although over smapling is a valuable approach, it must be applied after the cross validation splits in the pipeline; otherwise, we risk overfitting the model. Due to excess sampling, this frequently occurs in the situation of bootstrap sampling, where the same occurrences appear in both the training and validation sets. The test and validation sets, in any event, should be left alone. This step in the data analysis process is illustrated in Figure 3.8.

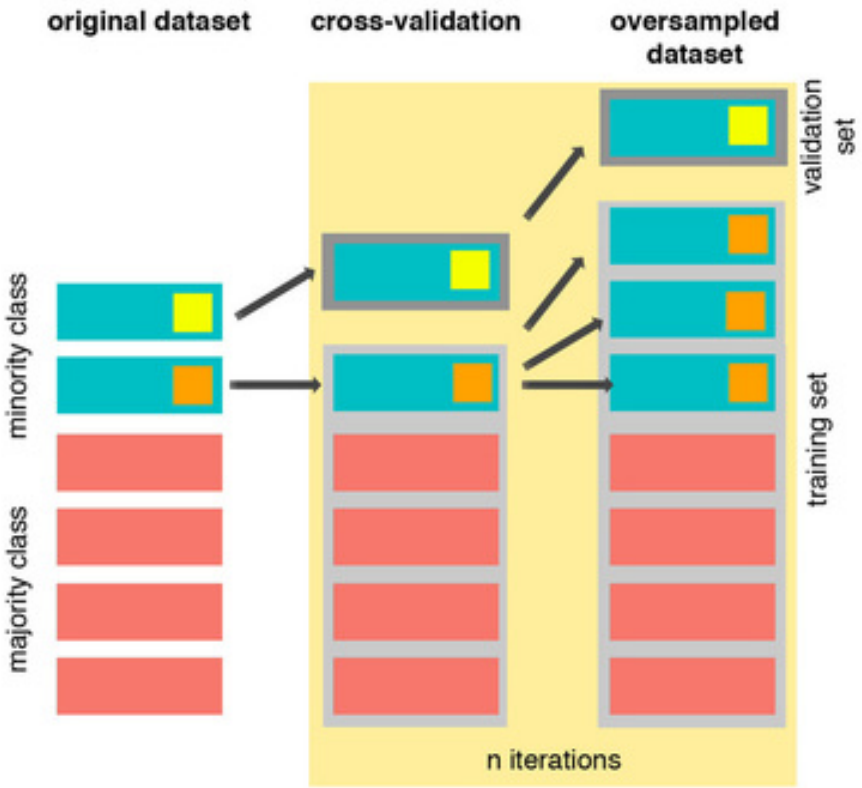


Figure 3.8: Sampling and Cross Validation

Chapter 4

Model Selection

4.1 Decision Tree

A decision tree is a machine learning technique that is simple to use. The goal is to build a model that predicts the value of a target variable by learning basic decision rules from data attributes..

4.1.1 Advantages

- The model is highly interpretable. Furthermore, this method does not necessitate any preprocessing, such as data normalization or label encoding. Missing values, on the other hand, must be dealt with.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.

4.1.2 Disadvantages

- Overfitting is an issue that arises when decision-tree learners generate overly complex trees that do not generalize the data well.
- If some classes dominate, decision tree learners build biased trees. It is consequently advised that the dataset be balanced before using the decision tree to fit it.

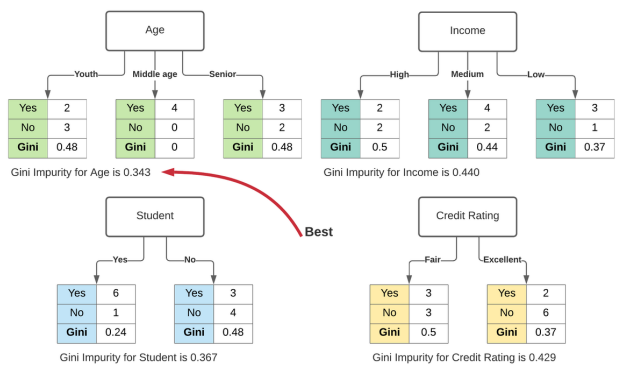
- Heuristics like the greedy method are used in decision-tree learning algorithms to make locally optimal decisions at each node. Such algorithms can't promise that they'll give you the best decision tree globally

4.1.3 Splitting Criteria

At each step, the best predictor to split is chosen based on a set of criteria. These metrics describe a split's purity. The most typical are:

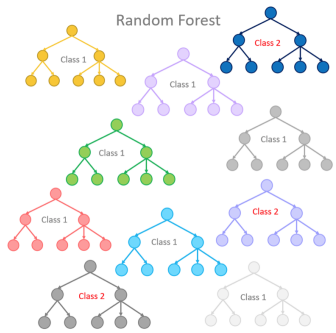
- Gini index: $1 - \sum_{i=1}^{n} (P_i)^2$
- Cross Entropy: $-\sum_{i=1}^{n} P_i log_2(P_i)$

Where Pi signifies the chance that an element will be classified into a specific class. Here's a visual representation of gini.



4.2 Random Forest

The ensemble class includes the random forest classifier. A number of decision trees are trained, each using a different set of bootstrapped training data. Only m predictors (typically $\sqrt{p}$) participate in each split. As a result of the splitting, uncorrelated trees emerge, resulting in a more robust model. The final decision is determined by a majority vote of the trees involved. *max_features or n_features.*



4.2.1 Bootstrap aggregation

It generates data sets by sampling with replacement, a simple yet effective data sampling technique. The term "sampling with replacement" refers to the fact that some data points can be selected many times. Each decision tree is trained on distinct randomly picked features to further reduce the connection between individual trees. Max_features or n_features is a hyper-parameter that controls the number of features utilized for each individual tree.

As a result, each decision tree in a random forest is trained on a separate data set, as well as various features/columns (due to bagging). Individual decision trees are trained separately before being joined. Maximum voting is utilized for classification. That is, the Random Forest's output will be the class, which is the result of most trees. The outputs of the Decision Trees are averaged for regression.

4.2.2 Hyper parameters

In order to tune the classification model, it's vital to understand the role of the hyper parameters in random forests.

- **Number of Estimators:** The number of trees in the forest.
- **Split criteria:** The function for determining the quality of a split (for classification, Gini impurity or Entropy; for regression, Mean-Squared-Error (MSE) or Mean-Absolute-Error (MAE)).
- **Maximum depth:** The maximum depth of a tree.

- **Minimum samples per split:** The minimum number of samples required to split an internal node.
- **Minimum samples per leaf:** The minimum number of samples required to be at a leaf node.
- **Maximum number of features:** The number of features to consider when looking for the best split.

4.3 Logistic Regression

A supervised statistical technique for determining the probability of a dependent variable is logistic regression (Classes present in the variable). The logit functions are used in logistic regression to assist derive a relationship between the dependent and independent variables by estimating the probabilities or possibilities of occurrence. The logistic functions (sometimes called sigmoid functions) transform probabilities into binary values that can then be utilized to make predictions. The question arises as to why regression is called regression but it is used as a classifier. The solution is simple: logistic regression is an extended linear regression model that uses the same basic formula as linear regression but regresses for categorical outcome probability. Instead of fitting the best fit line like in linear regression, the Logistic Regression condenses the output of the linear function between 0 and 1.

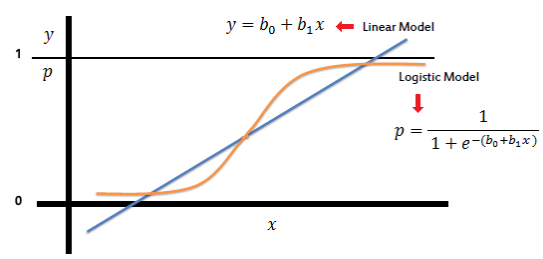


Figure 4.1: Logistic Regression

The function p is the logistic function, also known as the sigmoid function. In the formula of the logistic model, when $b_0 + b_1 x == 0$, then the p will be 0.5, similarly,$b_0 + b_1 x > 0$, then the p will be going towards 1
and $b_0 + b_1 X < 0$, then the p will be going towards 0.

As can be seen, the sigmoid function's range is limited to 0 and 1. We can now express data points as probabilities! We can now transfer each (x, y) pair from a set of points into a sigmoid function's graph and use it to calculate probabilities as predictions.

4.3.1 Decision Boundaries

When we feed the inputs through a prediction function that produces a probability score between 0 and 1, we anticipate our classifier to offer us a set of outputs or classes based on likelihood. We basically decide on a threshold value above which values are classified as Class 1, and if the value falls below the threshold, it is classified as Class 2.

4.4 Support Vector Machine(SVM)

A Support Vector Machine (SVM) is a discriminative classifier with a separating hyperplane as its formal definition. In other words, the algorithm produces an ideal hyperplane that categorizes fresh samples given labeled training data (supervised learning). This hyperplane is a line in two-dimensional space that divides a plane into two sections, with each class on either side. The Support Vector Machine, or SVM, is a linear model that can be used to solve classification and regression issues. It can solve both linear and nonlinear problems and is useful for a wide range of applications. The value of each feature is the value of a particular coordinate in this technique, which plots each data item as a point in n-dimensional space (where n is the number of characteristics you have).Then we perform classification by determining the best hyper-plane for separating the two classes. Take, for example, 2-D data with linearly separable classes that we want to separate. There may be several solutions to the problem, as indicated in the diagram.

Now that we have a variety of options, how do we decide which one to use? When employing a perceptron, all of the solutions were optimal because they separated the classes, but this is not the case with SVM. The SVM algorithm works by locating the hyperplane with the shortest minimum distance

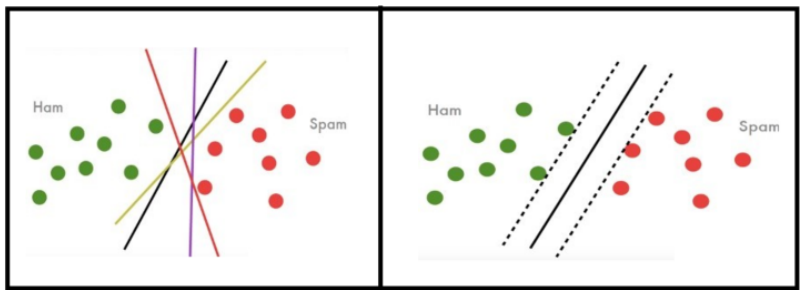


Figure 4.2: SVM separating 2-D data linearly

between the training instances. Within SVM's theory, this distance is given the crucial designation of margin twice. As a result, the best separating hyperplane optimizes the training data margin. The black line between Ham and Spam may be seen in the second photograph. The Maximum margin is the distance between the two dotted black lines.

It's possible that the data isn't linearly separable, for example.

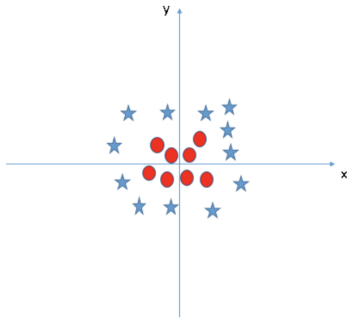


Figure 4.3: 2-D data linearly not separable

As you can see, there is no line that can be drawn between the two classes. However, with SVM, we can utilize kernel functions to turn data that is not linearly separable into data that is linearly separable. For example, using this simple kernel function, this data can be turned into linearly separable data.

$$z = x^2 + y^2$$

Using the given kernel we were able linearly separate data as shown below.

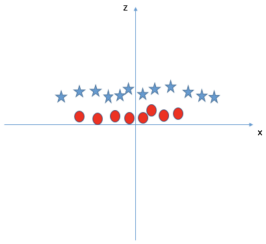


Figure 4.4: 2-D data linearly separable after using kernel function

We permit SVM to make a given amount of mistakes and keep the margin as large as feasible so that other points can still be categorized correctly when using real data. SVM's objective can be changed to accomplish this. As depicted in the diagram below.

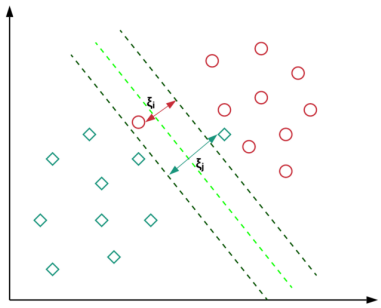


Figure 4.5: Linearly inseparable but SVM with tolerance

We used SVM on our data set after analyzing all of these properties, and the results are presented in the results section.

4.5 K-Nearest Neighbour

We frequently categorize people based on their proximity to the people they live with. People who belong to the same group are frequently grouped together based on their shared qualities. The KNN algorithm is based on the simple notion that "birds of a feather flock together.""
The supervised machine learning technique K-nearest neighbors is frequently employed in classification tasks. It is based on the simple premise that "apples do not fall far from the tree," implying that similar items are constantly near by. The data points are categorised using this method based on how their neighbors are classified. A similarity measure of all the available instances is used to classify any new example. Technically, the technique classifies an unknown item by looking at k of its already-classified, nearest neighbor items and calculating the majority votes from those with similar properties to those used to map the things.

- **Lazy Learning Algorithm**: It is a lazy learner since it does not go through a training phase, instead memorizing the training dataset. All computations are postponed until the classification is completed.
- **Case-Based Learning Algorithm**: The algorithm is known as an instance based or case-based learning algorithm because it makes predictions using raw training instances from the issue area. Case-based learning indicates that KNN does not learn a model explicitly. Rather, it memorizes the training cases/instances, which are then employed as "knowledge" during the prediction phase.
- **Non parametric**: It makes no assumptions about the distribution pattern of the underlying data..

4.5.1 Selection of K in KNN

The number K in KNN refers to the number of nearest neighbors taken into account while giving a label to the current point. K is a key parameter, and determining its value is the most difficult task when using the KNN algorithm. The process of selecting the appropriate value of K is known as parameter tuning, and it is critical for improving accuracy. If K is too little, the model may be overfitted, and if it is too large, the technique becomes

computationally expensive.
Choosing the value of K is a case-by-case decision, and sometimes the best way to choose K is to experiment with several values of K and compare the results. The KNN algorithm can be evaluated for several values of K using cross-validation, and the value of K that results in excellent accuracy can be regarded the ideal value for K..

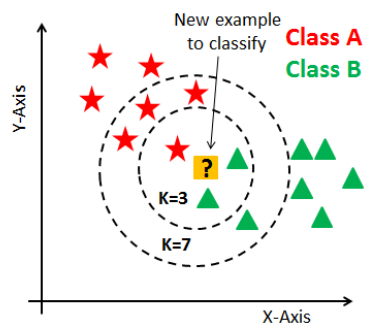


Figure 4.6: What K to choose?

As you can see in the graphic above, if we use K=3, we predict that the test input belongs to class B, and if we use K=7, we anticipate that the test input belongs to class A.

4.5.2 Determining a point as neighbor

We mentioned that we classify a point in a data collection according to its closest neighbors, but what exactly does it imply? In this context, it refers to how similar/close the data point is in distance to the training samples. The following formula can be used to determine distance.

- Euclidean distance
- Manhattan distance
- Hamming Distance

- Minkowski Distance

The objective behind using distance measure is to discover the distance (similarity) between a new sample and a training set, and then find the k-nearest samples to the new data point; all of these k data points are known as the k-nearest neighbors of that point.

4.5.3 Working

Working principles of KNN classifier is given in the points below.

- Choose a value for K. K should be an odd number.
- Find the distance of the new point to each of the training data.
- Find the K nearest neighbors to the new data point.
- Count the number of data points in each category among the k neighbors. New data point will belong to class that has the most neighbors.

KNN algorithm is a good choice if you have a small dataset and the data is noise free and labeled. When the data set is small, the classifier completes execution in shorter time duration. If your dataset is large, then KNN, without any hacks, is of no use. And in our case the data set was not very large and we also removed outlier using isolation forest hence we could easily afford to use KNN.

With k=5 and 4 of the yellow data point's neighbors belonging to M class and 1 belonging to L class, we'll assign M class to the new data point.

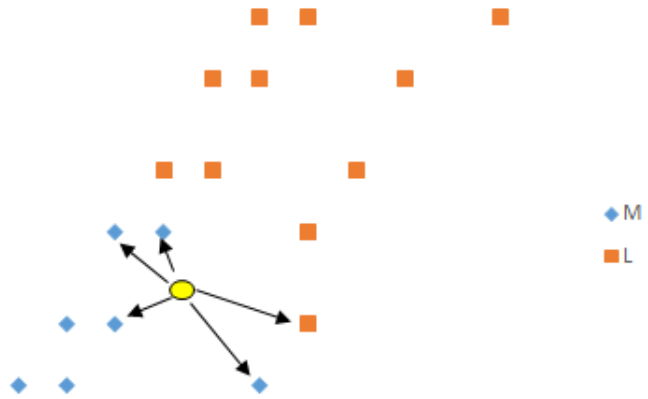


Figure 4.7: K-Nearest Neighbours classifier

Chapter 5

Evaluation metrics

Metric evaluations are performed in order to understand how well a classifier has performed. Mainly we use the following evaluation techniques:

5.1 Some Notations

Assuming that we have two classes A,B

- TP (True Positive) : when point belongs to class A and was predicted as class A. We say it is True Positive for class A.
- FP (False Positive) : when point doesn't belong to class A but was predicted as class A. We say it is False Positive for class A.
- TN (True Negative) : when point doesn't belong to class A and was predicted as class B. We say it is True Negative for class A.
- FN (False Negative) : when point belongs to class A but was predicted as class B. We say it is False Negative for class A.
- TPR (True Positive Rate) : Ability of classification model to identify all relevant instances.So when it is actually A,how many times it was predicted A.
- FRP (False Positive Rate) : When it is actually A ,how often it is predicted B.

5.2 Accuracy

Accuracy is the ratio of correctly predicted label with respect to the total number of instances in the test set.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

It's possible that the examination will lead to erroneous findings. When one or more classes in the dataset are underrepresented, this can happen. In a binary classification with classes A and B, for example, A is present 90

5.3 Recall

The ratio of anticipated true positives to total positive cases is sometimes of relevance. This metric is especially important when it comes to health and medical services. For example, one would be curious about how well the classifier performs in diagnosing a condition when compared to all of the true cases of disease.

$$Recall = \frac{TP}{TP+FN}$$

5.4 Precision

Another way to look at a classifier's findings is to look at the ratio of predicted true positive cases to total cases that were projected positive.

$$Precision = \frac{TP}{TP+FP}$$

5.5 F1-score

The F1 score can be thought of as a weighted average of precision and recall, with the best value being 1 and the poorest being 0. Precision and recall both make an equal proportion to the F1 score.

$$Precision = 2\frac{Precision*Recall}{Precision+Recall}$$

Chapter 6

Model Evaluation

The table describes the results of metric evaluation on the classifiers.

Metric	Decision Tree	Random forest	Logistic Regression	SVM	K-NN
Accuracy	0.75	0.79	0.75	0.75	0.78
Recall	0.56	0.61	0.55	0.55	0.60
Precision	0.72	0.74	0.69	0.69	0.69
F1-score	0.63	0.67	0.61	0.61	0.64

Bibliography

[1] Liu, F. T., Ting, K. M., Zhou, Z. H. (2008, December). Isolation forest. In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on (pp. 413–422). IEEE.