CAPSTONE PROJECT REPORT

Problem Domain

I have chosen Health Science Analytics as the problem domain.

In the healthcare industry, currently drugs developed which solve a problem in one part of the body sometimes creates problems in other parts. This is because different molecules in the body respond differently to the same chemical composition of drugs. To eliminate or minimise this problem a lot research is being done on targeted treatments, where developing a new drug is based on animal or human testing. This takes up a lot of time and money to arrive to the final chemical composition of that specific drug. By adding the flavour of machine learning, we can study every molecule to know how it responds to various stimuli.

The molecule will be provided with an external chemical stimulus and either it will respond do it or not. The 'biological response' we get will be specific to a molecule, i.e., some molecules will respond to it and some won't. Now each molecule has properties like shape, size, chemical composition, etc. It is these properties that may or may not cause a molecule to respond to the chemical stimulus. This needs to be studied because it can be used to predict whether a new molecule having similar characteristics would respond or not.

Problem Statement

The objective is to study molecular information provided in the dataset and using it make predictions whether different molecules would respond to chemical stimuli when provided.

The objective will be achieved by using machine learning. A 'model' needs to be developed which will solve this problem. This model will have algorithms to reduce the data and extract useful information and use it to study patterns in the responses gained by particular molecules. By providing sufficient data, the model will learn when biological responses are elicited and be able to predict in the future the possible outcomes of different molecules.

If successful, we should be able to predict biological responses of molecules, which in turn can be used to make drugs specific to a problem for a particular molecule. This analysis can provide better support to targeted treatments where we cure the patient's affected are only and not harm the remaining cells.

Performance Metric

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

Where

- TP: True Positive
- FP: False Positive
- FN: False Negative
- TN: True Negative

There are many performance metrics that can be considered, like: F1 score, log loss and accuracy score.

I did not consider accuracy score as it suffers from accuracy paradox. So even if the model created doesn't classify any of the molecules to have a positive response, even if they do have one, the score is still very high. So the model would actually be useless, but it will have a very good accuracy score.

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

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

Where

TP: True Positive

FP: False Positive

FN: False Negative

Instead we could use precision and recall which tells us how many positives the model was able to classify correctly and relevant results that it could retrieve, respectively.

		Prediction	
		0	1
Actual	0	TN	FP
	1	FN	TP

Where

- TP: True Positive
- FP: False Positive
- FN: False Negative
- TN: True Negative

Figure 1: Confusion Matrix

The confusion matrix was used to provide a more detailed analysis of the results. As shown in figure 1, the matrix tells us exactly how the algorithm is confusing the 2 classes by providing false negatives and false positives, along with the true negatives and true positives.

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

The F1 score is a balanced combination of precision and recall. I used the F1 score in the grid search for it to decide which algorithm was the better one and give us the final accuracy of the tuned model.

Log loss is commonly used in logistic regression to compare probability outputs of a classifier instead of finding discrete classes, we can find the probability of a data sample belonging to that class. Since it is a loss function, the lower it is, the better. So a value between 0 and 1 is ideal. For a binary classifier, the negative log likelihood for of the classifier given a true label is:

$$L_{\log}(y, p) = -\log \Pr(y|p) = -(y \log(p) + (1 - y) \log(1 - p))$$

Where

- p: probability estimate (Probability of true label)
- y: classes (0 or 1)

[Source: https://tryolabs.com/blog/2013/03/25/why-accuracy-alone-bad-measureclassification-tasks-and-what-we-can-do-about-it/]

Example:

True Positive (TP)	False Negative (FN)	Accuracy (ACC)	Precision (PREC)
5	1	**0.996**	**0.625**
False Positive (FP)	**True Negative (TN)**	**Recall (TPR)**	**F1 Score**
3	991	**0.833**	**0.714**
Positive events (Pos)	**Negative events (Neg)**		
6	**994**		
Positive observations	**Negative observations**		
8	**992**		
Total events (Te)			
1000			

Strategy

The model will be created by studying the molecular properties and their activity. There are many properties provided in the data and the first step would be to reduce these features to be able use the data.

First the data will be scaled down to 2 dimensions to get an understanding of what kind of data distribution we are dealing with. Next the original data will be reduced to an optimal number of features so that we can use it in the analysis.

This problem is one where we need to output '0' when a molecule does not respond to a stimulus, and '1' if it does. This is called a binary classification problem. To achieve this, we use a binary classification algorithm. This algorithm will first study and divide the molecular data based on its output using the reduced feature set. The output will not just be the which class a molecule would belong to, but also the degree of confidence of model when predicting a particular class.

I have considered Support Vector Machines and Decision Trees to create the model. The final algorithm will be chosen depending how they perform.

About the Data

The data is taken from a competition which was hosted on Kaggle.

The training and test datasets are present in the comma separated values (CSV) format. Each row in this data set represents a molecule. The first column contains experimental data describing an actual biological response; the molecule was seen to elicit this response (1), or not (0). The remaining columns represent molecular descriptors (d1 through d1776), these are calculated properties that can capture some of the characteristics of the molecule - for example size, shape, or elemental constitution. The descriptor matrix has been normalized.

- No. of Features: 1776
- No. of Data points: 3751 (training), 2501 (test)
- No. of Molecules elicited response: 2034 (54%)
- No. of Molecules that did not elicit a response: 1717 (46%)

The data seems to be evenly distributed.
Minimum, maximum and mean of every feature has been calculated and stored in 'Features Information.csv'

Data Exploration

A subset from the entire data was chosen (500 samples) to perform data exploration. These samples were reduced to a 2D space and visualised on a scatter plot. The X and the Y axes of the plot are the 2 features obtained after performing feature reduction. Figure 2 shows the resultant was a concentric elliptical distribution.

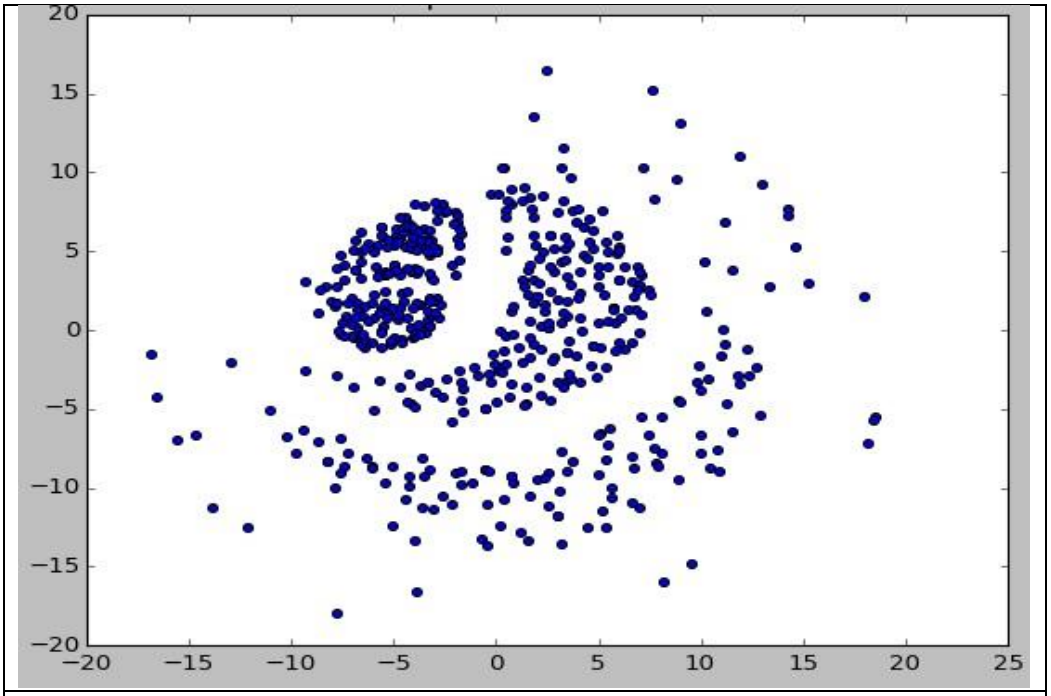

Figure 2: Data Distribution of randomly chosen samples

An elliptical distribution signifies a multivariate normal distribution. So if the linear combination of 1776 components would be visualised, we would be able to see a normal distribution of data. This can be seen in the histograms provided in figures 4 and 5 where the 2 features of MDS were visualised and we got normally distributed data (skewed).

[Source: https://en.wikipedia.org/wiki/Elliptical_distribution]

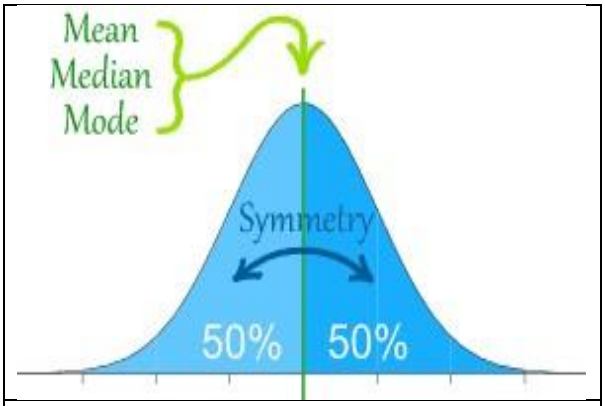

Figure 3: Normally Distribututed Data

Figure 3 demonstrates a Normal Distribution which is a continuous valued symmetrical graph having a central peak at the mean.

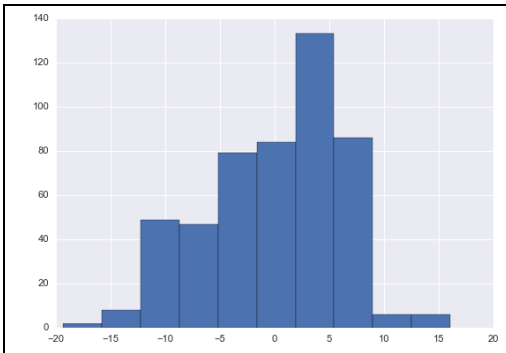

Figure 4: Feature 1 of MDS Reduced Data

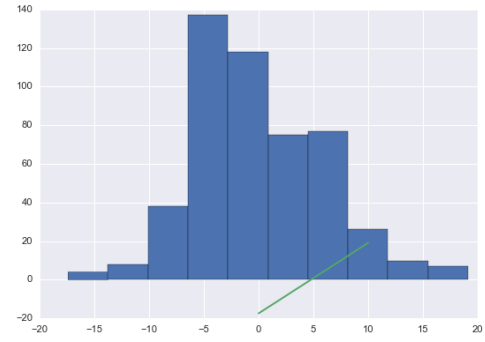

Figure 5: Feature 2 of MDS Reduced Data

Data Sampling

This is a technique which chooses a small subset of the entire population (the molecules data set) to represent it. I used the sklearn.utils.resample method to pick 500 data points. It takes the entire dataset as its input along with number of samples required and if resampling with replacement would be preferable when performing 1 step of bootstrapping procedure. It outputs a resampled array having 500 data points.

Feature Reduction for Exploratory Analysis

The data present has 1776 dimensions, which cannot be visualised successfully on a 2 dimensional graph. Thus, the features are reduced to a 2 dimensional array using the sklearn.manifold.MDS method. One of suggested techniques for visualisation was Multidimensional scaling (MDS). MDS models the similarity of data by maintaining data point distances of higher dimensions while mapping it to lower dimensions. This way data distribution patterns can be easily identified and we can choose appropriate algorithms to analyse the data.

[Source: https://en.wikipedia.org/wiki/Exploratory_data_analysis]

Visualisation

The visualisations were mainly done by matplotlib.pylot to display a scatter plot of the feature reduced data samples.

Pylab plots were used to display line charts to understand the performance of classification algorithms based on the parameters used. This helped to identify the range of best parameters.

Seaborn library was used to visualise heat maps for the confusion matrix. We can easily understand how many data points has the model misclassified or how well has the model correctly classified the data.

Classification Models Considered

On the basis of my analysis, the classification models SVM and Decision Tree seemed to be the best for this problem for the reasons given below:

SVC

Support Vector Machines, or SVMs are supervised learning models which are generally used for classification and regression. Also, SVCs or Support Vector Clustering can perform multiclass classification but the model is created by doing a one-to-one combination of classes.
SVC can be used for this problem statement as we have to perform binary classification of the data. The algorithm divides the given samples in such a way that there is a clear gap (as wide as possible) between the closest 2 samples of different classes, when more data points turn up, it is less likely to be misclassified.
Figure 6 depicts how support vector machines classify data.

a. Applications
   i. Anomaly detection (outlier analysis)
   ii. Dimensionality reduction
   iii. Classification of images, character recognition

b. Strengths
   i. Can perform non-linear classification
   ii. Effective in higher dimensions even when number of samples are lesser than the dimensions

c. Weaknesses
   i. Parameters of a solved model are difficult to interpret

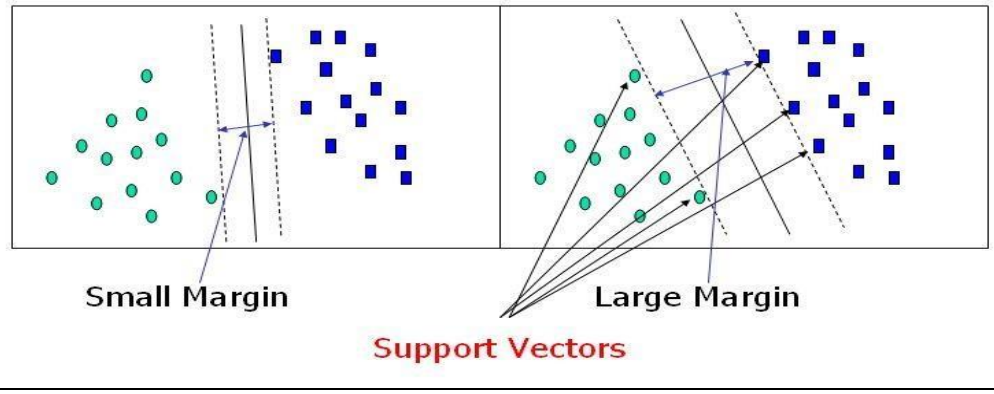

Figure 6: Support Vector Machines

Decision Tress

Decision Trees, or DTs that uses a graph or tree like structure to map decisions with consequences and then its consequences and so on. The tree has nodes, which are the decisions or consequences and edges or paths which lead to consequences or a particular decision as shown in figure 7.

Every DT has only 1 root node which is the starting point of the algorithm. This node splits into a number of branches based on the number of decisions it can possibly take, in our case, it will be some optimal number of features. Each node keeps splitting and producing child nodes to finally give us an answer or a number of decisions taken. Every decision taken in 1 path leads to a leaf node which has our final decision.

DT makes it much easier to understand how the algorithm arrived at that particular step. Also, for every scenario we may get to know what the result would be since DTs explore every option / path.

a. Applications

i. Decision analysis to identify strategies

ii. Astronomy

b. Strengths

i. Easy to interpret

ii. Allows addition of new scenarios

c. Weaknesses

i. Tend to overfit the training data

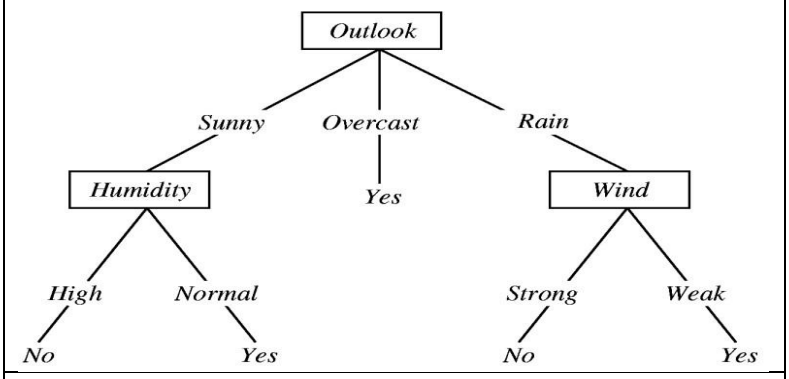

Figure 7: An example of a Decision Tree

Benchmark

Prediction probabilities for every molecule of test set was provided by Kaggle in the 'benchmark.csv'. These values are the confidence by which the model predicts that a particular data point belongs to class 1. The model created will be calibrated to output classes and prediction probability of those classes and will be compared to this benchmark.

The public leader board on kaggle has provided positions of contestants in ascending order of the log loss value of their classifier. I'm going to set the benchmark for my classifier to be better than the 10^{th} rank, i.e. 0.40870.

Analysis of Models Used

Data samples were taken and visualised in a 2D space, by which I concluded that I have a multivariate normally distributed graph.

Data visualisations for PCA reduced datasets were done for both the models with various parameters in each case.

In case of SVC, I varied the value of the C parameter from 0.01 to 1000. I generated a graph mapping f1 scores to values of C. As C increases, the F1 score for training set improves but he test set becomes gradually worse. Thus we need to find a meeting point where the errors are balanced. That would be somewhere between 0.01 and 100 with F1 score around 0.8 to 0.85. These values are then fed to the grid search to find the best parameter.

With decision tree classifier, I varied that max depth from 1 to 20. A graph mapping the F1 scores to depth was generated. The trend observed is similar to the case of SVC. Also I assume the meeting point to be around 2-10. This range will be fed to the grid search to finding the best depth for the given data.

Feature Transformation

The dataset has 1776 features which describe a molecular response. Though having more features makes the data more descriptive, it is infeasible to use all of it and in most cases, not all the features are useful. PCA discovers dimensions from the data that best explain the variance within it and creates new features which are combinations of the original many features. Thus, we can reduce the number of features by using PCA transformation.

Tuning Parameters

Pipelining was used to perform a 2 step grid search for the classification models considered above. The results from the estimators are assembled and cross-validated together to give the best results for different parameters.

The first step involves finding the minimum number of features that we can use to perform the classification of data. Here we find the best value for n_components (number of features) for PCA dimensionality reduction.

The second step involved finding out the max_depth for the decision tree or the C value for SVC.

This way 2 pipelines were created and fed to GridSearchCV algorithm twice for finding the best estimators as shown in figures 8 and 9.

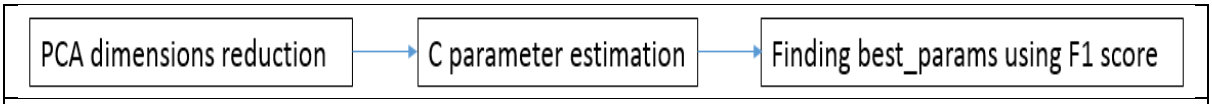

Figure 8: Grid search pipeline 1 to find best parameters using SVC

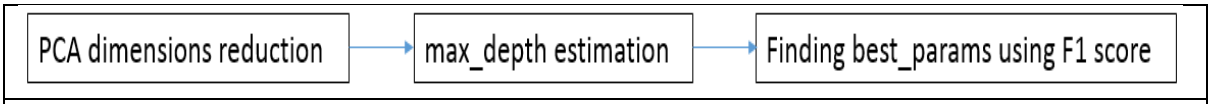

Figure 9: Grid search pipeline 2 to find best parameters using Decision Tree

Transforming the Training Data

PCA component analysis was used to reduce the number of dimensions for the analysis. Using GridSearchCV, number of features was reduced.

GridSearchCV Performance:

SRNO	Classifier	Parameter	PCA Best n_components	Best Classifier Parameter Value	Time taken (in minutes)
1	SVC	C	200	10.0	20.8

2	Decision Tree	Max_depth	150	4	17.23

Training and Predicting

The PCA transformed data (training data) was used to train the Decision Tree and SVM classifiers using the best parameters calculated by Grid Search. The program was run multiple times and SVC gave a better F1 score than decision tree.

The table below gives best_params outputs of the grid search for SVC

SRNO	N_components	C	Best_params PCA	Best_params SVC
1	100, 300, 600	1.0, 10.0, 100.0	100	10.0
2	50, 100, 150	1.0, 10.0, 100.0	150	10.0
3	100, 150, 300	1.0, 10.0, 100.0	150	10.0
4	150, 200, 250, 300	1.0, 10.0, 100.0	200	10.0
5	220, 230, 235	1.0, 10.0, 100.0	235	10.0

The table below gives best_params outputs of the grid search for Decision Tree.

SRNO	N_components	Max_depth range	Best_params PCA	Best_params max_depth
1	100, 300, 600	1 - 10	100	4
2	50, 100, 150	1 - 10	150	4
3	140, 150, 170	1 - 10	140	4
3	140, 145, 150	1 - 10	145	4

Decision trees tend to overfit the training data and perform really well on it, but when it comes to the test data, it performs rather badly. We need a model that has the ability to generalise to a degree and yet by able to give accurate predictions. Hence I decided to use SVC.

Performance on Training Set:

SRNO	Classifier	Time taken (in minutes)	F1 Score	Log loss score
1	SVC	44.4	0.892	0.3198
2	Decision Tree	27.91	0.761	0.5588

Results

Log loss was used to compare the benchmark probabilities with model prediction probabilities for the training set. Being better than the benchmark defined, we can say the model has done well.

Predictions made on the test set is were saved in 'Predictions.csv' file once they were complete. Also probability of the classes assigned to each data point is saved in 'Predictions Probabilities.csv'

Performance on Test Set:

SRNO	Classifier	Time taken (in minutes)
1	SVC	0.04

The prediction probabilities made were compared to the benchmark values provided graphically. The graph shows that to some extent, the model has been able to fulfil the objective.

Conclusion

Reflection

The data sets provided in the csv files were loaded, divided into test and training sets and were prepared to be used for analysis. Some statistical analyses were done to understand the data distribution which was helpful in choosing the classification algorithms.

Sampling and its visualisation was particularly helpful to be able to decide what range of parameters to pass in the Grid Search algorithm. Use a wide range of parameters with such a large dataset would not have been feasible. On the basis of this output I then selected a much smaller range of values to tune both the classifiers (SVC and Decision tree) and find the best parameter for feature reduction using PCA, which took much lesser time than it would have, had I not done sampling.

The tuned classifiers were then fit to the training set of train.csv file provided to create 2 models and tested against the test set of train.csv. I used the optimal 3 fold cross validation to achieve the best model, which was SVC.

The training and testing score were plotted for various values of C [0.01, 0.1, 1.0, 10.0, 100.0, 1000.0] for the SVC classifier. The increase in value of C means the increase in model complexity. Initially both the training and testing F1 score increases as the model fits the data better. But after a point training score improves to more than 0.95 but testing score keeps declining and goes below 0.8. This is because the model becomes overly complex and 'overfits' the data. So it performs very well on training set, but when provided with a new dataset having different points, it is unable to correctly classify those data points.

This pattern can be seen in figure 10. Thus we need to find a meeting point such that F1 scores for both training and testing set is balanced and according to GridSearchCV, that is when C = 10.0.

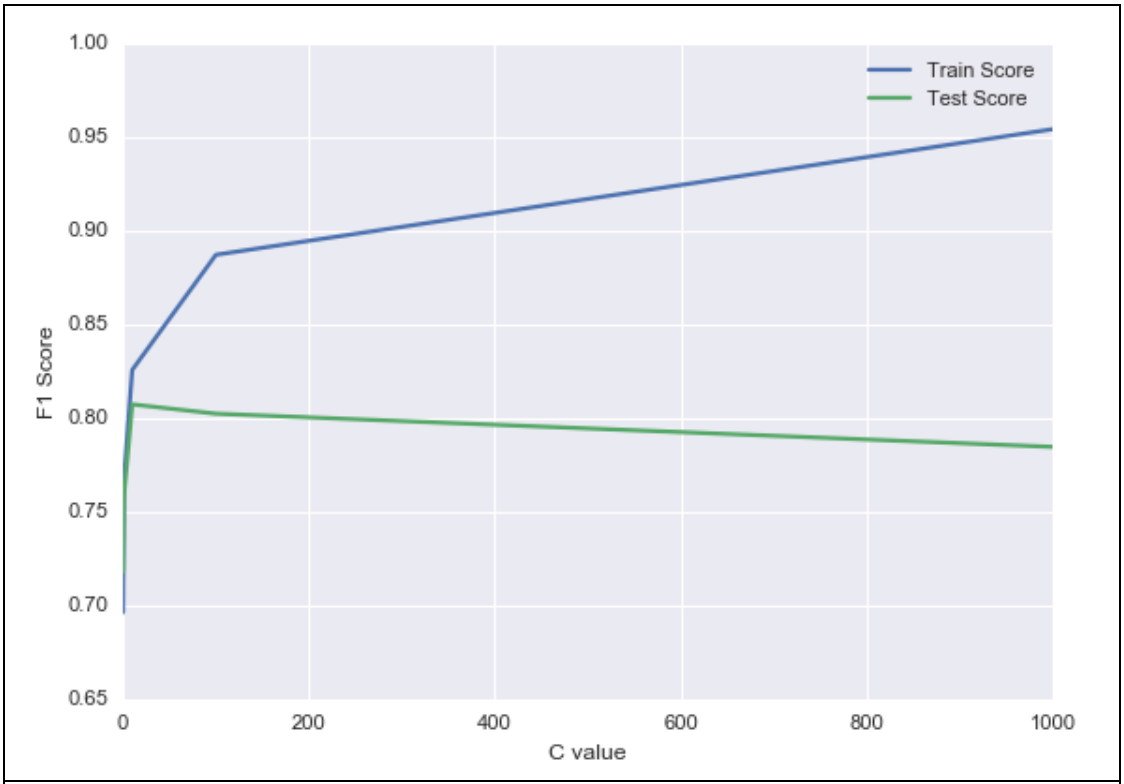

Figure 10: SVC Training versus Test Score for values of C

The model has successfully analysed the training data provided and correctly classified almost 90% of the data points. It has learnt about molecules and what properties of a molecule will cause it to respond to a chemical stimulus. Then this was applied to the test set and predicted the probability of classes a data point would belong to. Figure 11 shows, when the probabilities of 100 random samples were compared to their benchmarks provided, there were very little deviations from it.

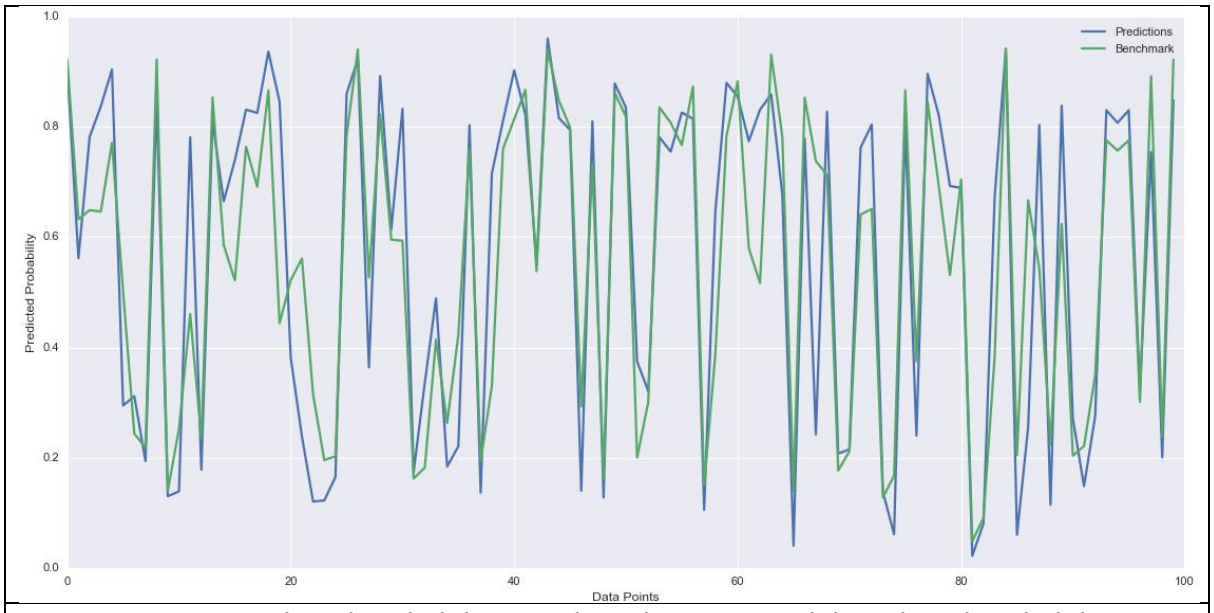

Figure 11: Predicted Probability Benchmark versus Model Predicted Probability

The confusion matrix shown in figure 12 explains how well the SVC model has classified the data points of the entire training set.

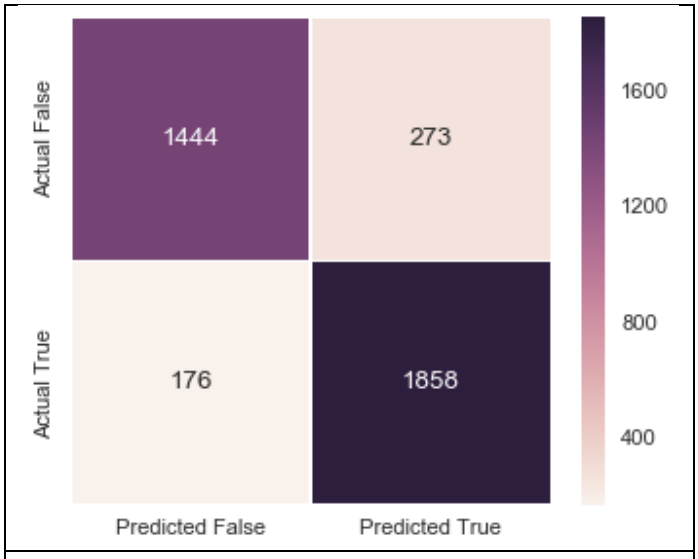

Figure 12: SVC Confusion Matrix

The matrix represents how accurate the SVC model is. It correctly predicted 1444 'no response' data points out of the total 1717 actual 'no response data points, which is an accuracy of 84%. And of the 2034 'responded' data points, it predicted 1858 correctly, which is 91% accurate (recall). If we consider precision, this model has precision of 0.872.

The model has an F1 score of 89.2% which is fair but can be improved.

Since SVM had a better F1 score in the visualisations on the test set, the predictions made on the actual unlabelled set can be relied on.

Improvement

Since the SVC is using RBF (radial basis function) kernel to create the classifier, we changed the C parameter but kept the gamma parameter constant. Gamma decides who influential a particular training point can be while creating the model. Low values translates to training points having a larger area of influence, which is the inverse of the area of a particular sample selected by the support vector. If gamma is very large, it would mean it covers only itself (own support vector) and area of influence is small which would to overfitting. Similarly, if it is too small the area of influence will cover more points in a training set and the model will be unable to capture the required complexity by the classifier. This way, gamma and C work together to prevent overfitting while capturing as much complexity as needed.

Thus a classifier is very sensitive to gamma value, even a slight tweak in it can either make the model much better or much worse. So the model can be improved by seeing what value of gamma should be used with C in order to increase the F1 score of the classifier.