COMP30027: Machine Learning Assignment 2: Report

1. Project Overview

   The task is related to multiclass classification of duration labels indicating the time it takes for a particular meal to be prepared.
   In this task, we deal with 2 numerical and exactly three text related features. Since text features can't be given as input to classifiers, we use certain machine learning tools namely countvectorizer , which converts text into bag of words with their relative frequencies and Doc2Vec which converts text in vector format that becomes extremely handy for classification/regression as one can compute distances with vectors.

   For this task, we start of by some feature engineering making an effort to select only those features that contributes to the outcome and trying to reduce the complexity of our classifiers, After which different types of classifiers are used, along with some form of hyper parameterization optimization to get an increase in accuracy which is then coupled with the use of an ensemble model to get accuracy better than all individual classifiers.

2. Feature Reduction.

   To start with we deal with 2 numerical features, and using doc2vec 50, each of the text features where converted into vectors of size 50 with total samples being 40000, therefore in this case we have 152 features in total.
   Now making sure that we have all relevant and no redundant features is very important since this leads to what is known as curse of dimensionality. Therefore mutual information is used to determine which features are more important than others since the target variables are discrete.

   Intuitively what mutual information calculates is basically the relation between the independent variable (feature) with the target variable. So a value of zero implies that there is no relationship between the feature and the target variable and vice versa. In Simpler words it can be said that mutual information of zero implies that knowing about one random variable does not tells anything about the other random variable. Mutual information can be calculated by the following formula:$I(X;Y) = H(X) - H(X|Y)$ where $H(X)$ is entropy of Variable X and $H(X|Y)$is conditional entropy of X given Y

   So from the given data set we get the following results:

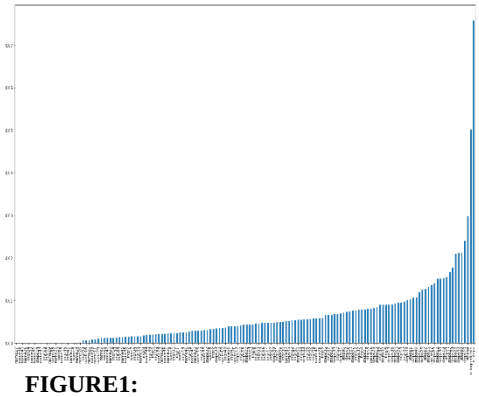

FIGURE1:
mutual
Information of all features.

N_steps	0.073692
N_ingredients	0.054711
Steps4	0.029799
Steps10	0.024005
Steps8	0.021241
	…
Name19	0.0000
Steps38	0.00000
Name14	0.00000
Name13	0.00000
Name47	0.0000

Table 1: indicates MI of all features and gap In between indicates all other features.

From the table one can see that there indeed are many features that have MI of zero implying that these features are independent and therefore does not express much about target variable.

Note that MI is indicating relationship between independent and target variable, and hence for MI = 0, one can remove that particular feature. This in not to be confused with relation between different features.

For the rest of the project threshold for MI is considered to be 0.001. The reason to do so is to reduce the complexity and computational cost of some classifiers used later.

3. Classification using Stacking.

IN this project, the method implied for obtaining accuracy is by using stacking.
In simple words stacking uses multiple classifiers at base to obtain predictions and then using another classifier at final step to make final predictions.
The final model in stacking is trained on a dataset of predictions obtained from the classifiers used at base.

Now let's say that our input data is $D(x_i, y_j)$ where
$i \epsilon [1,152]$ & $j \epsilon [1.0,2.0,3.0]$
Now after Feature reduction using MI we are left with only 124 features. Therefore $i' \epsilon [1,124]$.
Now we select classifiers at base level, namely Logistic regression, SVM and gradient boosting classifier lightgbm.

Each of these classifiers train on our data set and make predictions: Let $\widehat{y_j} = h(x_i)$ where $\widehat{y_j}$ is the prediction by the j_{th} classifier in our base and $h(x_i)$ represents transformation of all x features.

Next we create another dataset $D'(h(x_i))$ which is simply done by concatenation of all predictions column wise. Then our classifier at the level 1 is trained on this new data set along with the same target variables. The prediction made by the classifier at level 1 then becomes our final predictions. Note that it is not necessary to have one 1 extension of level. It can be many classifiers at various

levels leading to the classifier at the final level. The main reason of using stacking is to obtain accuracy better than the single classifiers.

In our project we use Logistic regression, SVM and light gbm at base and logistic regression at level 1 again.

First base model used is Logistic regression. This is a binary classifier that actual uses sigmoid function to make predictions. Now to emphasise on the importance of sigmoid function let's consider that instead of sigmoid function we use straight best fit line: $w^T x + b$. Assuming line passes through origin b = 0.

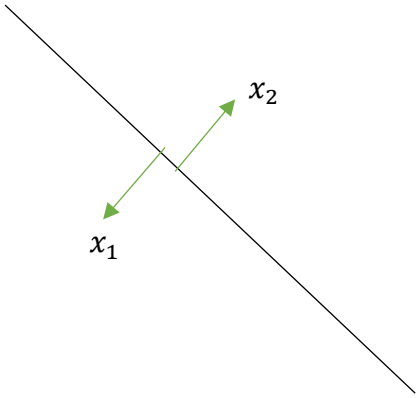

Figure2: Hyperplane intuition (LR)

Consider the above diagram and just for now assume that we have binary classification problem. Then distance from point to line is $w^T x$. From this we can see that all positive points above will have a projection line $w > 0$ and negative points will follow $w < 0$. Therefore all positive points will have $w^T x > 0$ and vice versa for negative points.

Now to this classifier makes predictions based upon $y * w^T x$. So for positive point y > 0 and therefore $y * w^T x > 0$, similarly for negative points $y, w^T x < 0$, hence their product will also be positive, which implies that a point is classified correctly. If not then we will get $y * w^T x < 0$. Therefore in LR aims to maximise $\sum y_i w_i^T x_i$ to find line of best fit. However this can be very volatile to outliers resulting in wrong decision boundaries just because of few points. Therefore we consider $f(y_i w_i^T x_i) = \frac{1}{1+e^{y_i w_i^T x\_i}}$, which is actually sigmoid function and it ranges from zero to 1 thus not being sensitive to outliers. Now for multiclassification case, LR consider one vs rest rule. In our case it considers 1.0 and 2.0 as positive and 3.0 as negative. Then at another level 3.0 and 2.0 is considered same class and 1.0 as different. All possibilities are enumerated and final predictions are based upon probabilities.

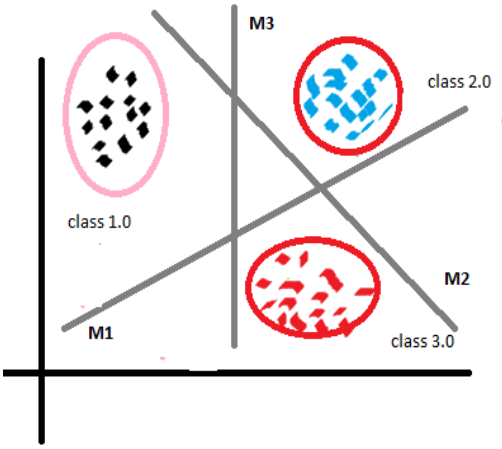

Figure3: diagram to illustrate the idea of linear sepration

LR does this by making separations as indicated by lines M1, M2, M3.

Now assume that the figure above depicts the distribution of classes 1.0, 2.0, 3.0. Then in this case LR will perfectly classify each one of them and hence our accuracy will be 100%.
However using PCA (n=2), we can see the following distribution:

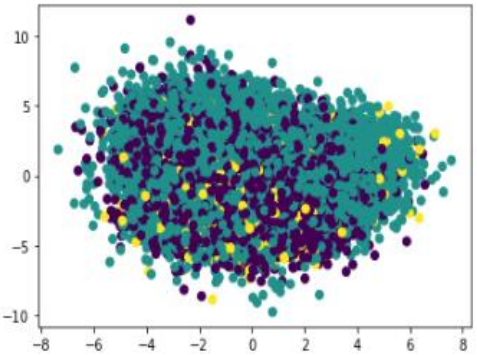

Figure 4: PCA (n=2) for visualization

Which implies that some points will not be linearly separable, then in such instances LR can make classification problems and hence this is depicted by an accuracy of 0.711400. Hence LR was unable separate classes perfectly.

Lastly, the hyper parameters used for LR are "ovr" that allows for multi class classification and "saga" as loss optimization algorithm since it converges fast with scaled data thus reducing computational complexity.

Moving onto our next classifier SVM. SVM have a very similar mathematical intuition to Logistic Regression. However, instead of finding best hyper plane by using the distance between point and hyperplane, SVM introduces margins that are also hyperplanes parallel to the main hyperplane. And optimization is considered by taking maximum of the distance of 2 margins. The further apart the two margins the better predictive (decision) power of the algorithm.

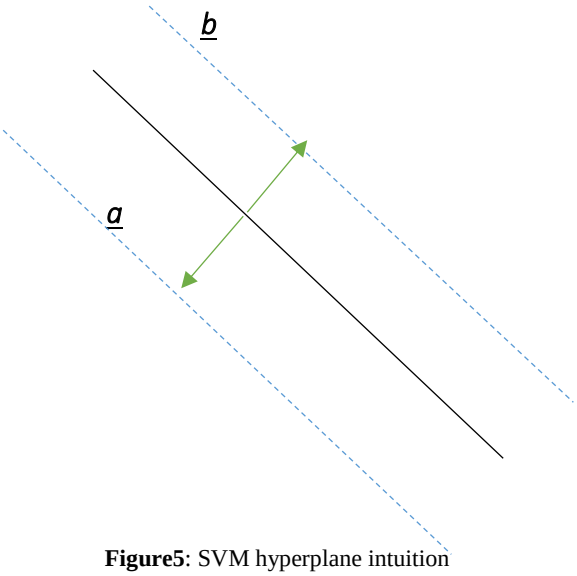

Figure5: SVM hyperplane intuition

The figure above illustrates what was mentioned above, solid line represents the hyperplane and dotted line represents margins. The key is to maximize the distance between margins which separates positive and negative points. Therefore margin **a** can be represented by the equation $w^T x_i + b = +1$ and **b** can be represented as $w^T x_i + b = -1$, where +-1 indicates the separation of Y (dependent) variables.
Now assuming that x_1 is a point on **a** and x_2 is a point on **b.** then distance between 2 margins is just $(w^T x_1 + b = +1) - w^T x_2 + b = -1 = w^T(x_1 - x_2) = 2$. So distance$(x_1 - x_2) = \frac{2}{||w||}$. We max on this distance such that $y_i * w^T x_i + b \geq 1$.

One can think of this as an updated version of Logistic Regression, however the idea of kernels in

SVM, makes it extremely powerful classifier.
One should keep in mind that in practise more often than not it is not possible to linearly separate 2 different points, this is where kernels are very useful. These can be used to map features on higher dimensions, where they might become linearly separable. These kernels include, polynomial, radial basis function and sigmoid function.
For example consider set of points on x_ axis only representing 2 different features.

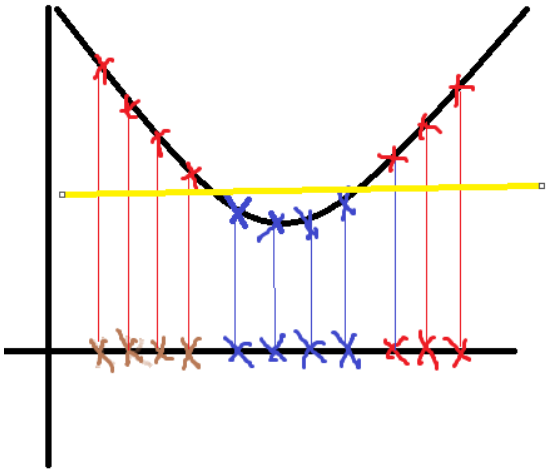

Figure6: example ploy (n= 2) transformation

From the above diagram one can see that two we cannot linearly separate two features red and blue, but mapping it onto a polynomial degree 2 curve, we can easily separate the points linearly.

Now from our data we can also see this kind of visualization:

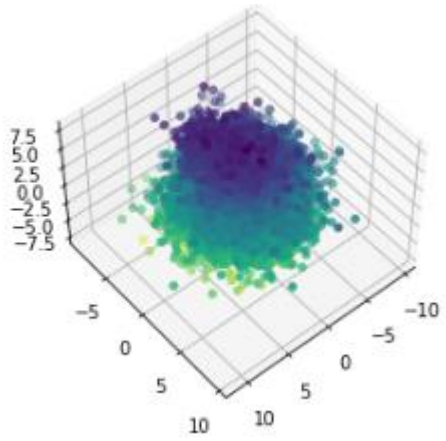

Figure7: PCA (n=3) for visualization

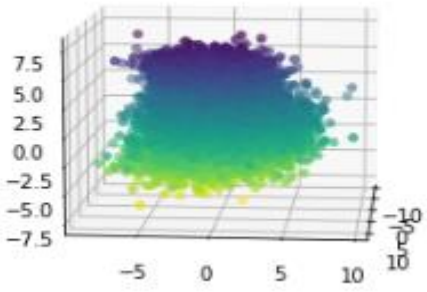

Figure8: PCA (n=3) for visualization

Earlier the 2dimensional plot clearly shows that many points are overlapping and hence are not linearly separable, but in 3 dimensions, it is visible that all yellow points cluster at bottom and blue ones at top, so not perfectly linear but still better than 2 d plot. By using “rbf” kernel we got an accuracy of 0.724650, which is better than LR, but still indicates that SVM was not fully able to produce hard margins, hence we have errors resulting to lower accuracy. Also not that LR and SVM produced almost similar accuracy which implies that class labels in 124^{th} dimension are partially more separable than first and second dimension that are shown, hence we get accuracy >> 0.5

Lastly we used light gbm that is gradient boosting algorithm. The main reason to select light gbm instead of other gradient boosting method is because of its computational efficiency. Unlike other gradients boosting methods, light gbm make leave wise growth rather them level wise growth and this results in less calculation time.

The idea of gradient boosting method is to minimize residuals. The way it does it is by making decision trees. So for instance in case of classification, the classifier starts off with initial prediction by the help of sigmoid function and log of odds. Next residual is calculated which tells us how bad our initial prediction. Based upon these residual we make further branches for decision trees and the output values are for the leaves are calculated as following

$$\frac{\sum Residual_i}{(previous\ probability_i * (1 - previous\ probability_i)}$$

And new log(odds) predictions = learning rate*output values.

The new predicted probability is then calculated that is nothing but substituting the value of new predicted probability in the sigmoid function.
We start again by calculating residuals, and continue until we have reached max number of trees specified or the residual is less than some threshold.
The intuition of binary classification is taken on to multiclass classification by using the same approach one vs. rest.

Note that learning rate is very important. If it's too large, our function might not converge and if it's too small converging can be very slow.

In our case using light gbm gives us an accuracy of 0.718.

4. Final steps
   Once we are done with calculating predictions, we pass on these prediction again to the logistic regression in level 1, which then predicts output based upon these input features and this lead to an increase in our accuracy to 0.7305

   Note that stacking can lead to overfitting and therefore it is very important to use cross validation method to avoid the problem of overfitting, so that our model can generalize well to other tests data.

5. Conclusions.

   To summarize, it can be stated that for the given dataset, it was very difficult for different classifiers to separate them linearly and therefore resulted in low accuracy (weak learners), to counter this problem, one can see that stacking can give a boost to overall accuracy. In the above case accuracy might not have risen extra ordinary but, this is because the stacking model above is fairly simple. With the inclusion of more individual learners and more learners with different hyper parameters and possible by adding more levels to stacking, it can be possible to achieve better accuracy.

6. References

Note that the recommended citation style for this report is defined as Harvard style, but you may use other (formal) citation styles if you prefer.

Majumder, B. P., Li, S., Ni, J. & McAuley, J. Generating personalized recipes from historical user preferences. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019.