MINI PROJECT -2 (INTRO TO AI)

EZHILMANI AAKAASH (40071067)

EXPERIMENTAL SETUP

1. ALGORITHMS

In order to obtain better experimental results and to understand the working of various Machine Learning algorithms, I took the liberty of working with 4 different ML algorithms.

1. Decision Trees
2. NaiveBayes Classifier
3. Support Vector Machines (SVM)
4. Random Forest
5. Neural Networks

As for the HyperParameters, I have used **2 different sets of parameters**. This is because, the two Datasets are different in terms of target label.

2. PARAMETERS

- **Decision Tree**

  tree.DecisionTreeClassifier(criterion="entropy",min_samples_split=2,max_leaf_nodes=None,max_depth=None,min_samples_leaf= 1)

  tree.DecisionTreeClassifier(criterion="gini",min_samples_split=20,max_leaf_nodes=None, max_depth=None,min_samples_leaf= 1)

  Criterion - A mix of both as the duo aren't really different from one another except in terms of computational speed.

  Min_Samples_Split - This helps tune the tree by providing the min samples required for split.

  Max_Leaf_Nodes – No of samples required at the leaf.

  Max_Depth – To provide the level of search (To limit over fitting).

  Min_Samples_Leaf -

- **Naïve Bayes**
  clf = BernoulliNB(alpha=0.5, binarize=None, fit_prior='True')
  clf = BernoulliNB(alpha=0.001, binarize=None, fit_prior='True')
  Alpha – Smoothing factor**.**
  Binarize – Since the given data is already binary vectors, set to None.
  Fit_Prior – Helps learn prior class probabilities.

- **Support Vector Machines**
  clf = SVC(C=10, gamma=0.001, kernel='rbf')
  C – Penalty Parameter
  Gamma – Parameter to control and adjust the points in higher dimension.
  Kernel – The mathematical function being used (Radial Basis, since no prior knowledge of data)

3. MACHINE LEARNING MODELS

- ML Algorithm – **Support Vector Machines**:
  This algorithm provides support for both classification and regression and depends on Supervised models. This model partitions the dataset using parallel lines into each categories in one or more high dimensional space. It supports both linear and non-linear classification which is distinguished using the "Kernel" parameter. The gap in-between the two classes represents the distance between the nearest two points of the two categories. Larger the seperation, lower the error.
  The reason I chose to work with this algorithm is because SVM performs data transformations based on the data we input and the flexibility of parameters. It also supports both linear and non-linear classification as mentioned. Despite the complex computations involved and slow performance, SVM provides stellar results for the given data set with low bias and variance.

- ML Algorithm – **Random Forest Classifier:**
  As the name suggests, Random Forest model tends to build a wide-variety of 'trees' or decision trees and merges them in such a way that the classifer/regressor provides great results. It also introduces a certain 'Randomness' while choosing the features to build over.
  The decision of working with this algorithm was concluded on the basis of 2 attribues.
  First and the most obvious one is the fact that it is an improvement over Decision Trees (One that I have used). Second, experimenting with this helped me understand more about decision trees and it's flaws.

4. ADDITIONAL CODE

- For experiemntation purposes, I also tried to work with Neural networks, a model that was taught in class. It had surpassed my expectations for both the data sets considering that fact that I was receiving poor accuracy for the Dataset-1 with other models. Therefore, I had modeled a neural network based on the given datasets.
- I have also written seprate set of codes for tuning the hyperparameters of each model.
- To have a better understanding of the mdoels, I came to the conclusion that besides the parameter tuning, a visual representation of the models returning results would be a better factor. Hence, I took the liberty of displaying the Confusion matrix and Classification Report.

ANALYSIS

As the focus for this project was more on the experimental side, I decided not to chase the best "accurate" model, rather try and understand how each works. This way, I get a better perspective of them and also try and improve on them.

1. Decision Trees

	ACCURACY – DS1	ACCURACY – DS2
BEFORE TUNING	28%	76%
AFTER – MANUAL	29.66%	76.87%
AFTER - GRIDSEARCHCV	31.9%	77.98%

- DT RESULTS

From the table drawn using the experiemntal results, it is safe to say that decision trees are a bad choice for dataset-1. One the other hand, for dataset-2, decision trees seem to achieve a better accuracy.
Upon examing the working of decision trees, I had discovered that these trees tend to "overfit" the data causing the predic() function to provide almost **100%** accuracy for training data while a bare minimum of **28%** for the validation data.

That being said, decision trees are not althogether bad at what they do. To elaborate more on that, I chose to work with an sklearn in-built dataset called the iris with a total of 150 samples. Working with it produced a staggering **98%** accuracy on most of the inputs for classification.

Diving back to our dataset, I tried to examine the dataset manually to get a better understanding. Upon doing so, I had identified that the DS-1 contains 1024 features and about 51 different classes to predict. Also, the **number of samples is close to the predictors**, hence producing a high-dimensional tree. This meant that the results heavily relied on the way the tree was built. With decision trees seeking the "best feature" to split, this is where the oveerfitting seems to occur causing such low accuracy.

First, I resorted to manual tuning of the parameters. The results are recorded above with hardly any signs to acceptable improvements.

In order to tackle the issue at hand, I chose to tune the parameters precisely, **MaxDepth** and **MinSamples**, using GridSearchCV.

MaxDepth meant the level of deepness the tree has to prune while MinSamples states the number of records required to make a split. This way, I believe that slight improvement achieved is because certain samples that were overfit earleir were handled. Despite that, this isn't a goof fit for Dataset-1.

While this isnt the case for dataset-2. This partly maybe because ds-2 has more entires allowing the model to beter understand the data and also because of the lower number of classes to predict (Ten)

2. Naïve Bayes

Following the bayes theory, naïve bayes assumes that the samples and predictors are independent. This way, the high variance that was witnessed in the Decision Tree causing low accuracy is reduced.

With the option of choosing from 4 different types, I chose BernoulliNB based on 2 factors. Theoretically, Bernoulli should be the best fit as it deals with binary input and multiple predictors. Also, just to be on the safer side, I tried fitting all the models. Here are the results.

	DS-1	DS-2
GaussianNB	43%	64.2%
MultinomialNB	49.8%	78.6%
ComplementNB	44.9%	62.2%
BernoulliNB	59.9%	80%

All of the classifiers perform way better than the decision tree for DS-1. This is purely because of the independence assumption it makes. The two classifiers that stand out are Multinomial and Bernouli. This is because of the ability to work on data with mulnomial distribution.

GaussianNB assumes the data is distributed in accordance to Normal Distribution and are continuous, which isn't exactly the case here, causing low accuracy.

ComplementNB, as the name suggests, estimates values based on all the classes except the one being evaluated. After viewing the test results, seems pretty clear that the algorithm isn't exactly that good of a fit.

After choosing Bernoulli, I began tuning the parameters using GridSeachCV.

BernoulliNB	DS-1	DS-2
Before Tuning	59.9%	80%
After Tuning	61.1%	81%

For DS-1, There seems to be a slight improvement after setting the 'Alpha' to 0.1 and taking previous probabilities. Since we are dealing with Bernoulli (binary data), we do **not** require a large smoothing additive. In case of Multinomial, Alpha of 1 or above showed signs of improvement.

In case of DS-2, there isnt much improvement. The idea of preprocessing did occur but the fact that the given dataset isn't too large and that it doesn't contain empty attributes meant that it did not require such. Upon fitting the data partially, **partial_fit(),** no improvement was recorded. This may be because the dataset is small in comparison to other large ones that take advantage of this feature.

3. **SVM**

Support Vector Machines help classify data using a hyper plane that tries to divide the attributes into classes. This works on both linear and non-linear distribution. The non-linear functionality is achieved using what's called the "**Kernel Trick".**

SVM	DS-1	DS-2
Before Tuning	50.5%	88.5%
After Tuning	66.9%	94.5%

With SVM, I was able to secure good accuracy even without having to tune the parameters. This is because SVM uses a strategy known as the "**Optimal Margin Gap**" that performs various transformations on data in 3-D hyperplane before it converts back to the original hyperplane.

Moving over to specifics, for DS-1, it could not match the performance of that of Naïve Bayes before tuning.
In case of DS-2, a solid 88.5% was achieved. This may correspond to the dataset having an optimal gap helping the model to better understand, train and process.

After tuning the parameters for SVM, a visible and large improvement in terms of acuracy was seen. The parameters tuned were 'C, Gamma, Kernel'.
param_grid = {'C':[0.001,0.1,1,10,100,1000],'gamma':[1,0.1,0.001,0.0001,1e-2,1e-4], 'kernel':['linear','rbf','sigmoid']} is the data used for tuning the parameters.

The major factor that seemed to have caused this push is the "Kernel Factor". Linear assumes the entire dataset to be completely linear while rbf or Radial function processes data in a nonlinear fashion in 3-D hyper plane.

Gamma refers to the parameters that decides the decision region. Lower the value, borader the region. 'C' refers to the penalizing criterion. If C is a small value, the classifier is fine with few outliers.

The only tradeoff I had to make was the computation time. While other models trained and provided the output well under 7 seconds, SVM needed about 50 to 60 sec approx. Considering the accuracy it provided, the trade-off seemed worthy.

4. **Random Forest**

I decided to experiment with random forest to get a better understanding of the flaws of decision trees. This way, I can work better with the other data sets and also get to know about few more algorithms/classifiers.

Random forest is a simple to use ML algorithm that requires less computation time but provides results that are of good accuracy even without having to fine-tune the parameters.

RF	DS-1	DS-2
DT- Tuning	43.5%	83.9%

Random forest is basically a culmination of multiple decision trees where the split is not made based on the "most important attribute" rather, based on the best feature **among the random subsets.** This way, the trees formed are of wide variety and different in structure.

This leads to a better understanding of the attributes and mainly reduces the act of **overfitting** which was one of the main reasons for the poor accuracy in case of decision trees.

My guess is that, if the random forest is more finely tuned, better results can be achieved.

5. **Neural Networks**

Known for their ability to "learn" and make adjustments, artifical neural networks are always a good choice when it comes to supervised learning. This is true because of the working structure of the model which resembles that of a neuron consisting of perceptrons.

In order to test it's functionality, I tried running ANN for the given datasets.

ANN	DS-1	DS-2
Without Tuning	64%	90.25%

In comparison to other models discussed, ANN seems to be way ahead of them even without having a tune a single hyperparamaeter.

The paramater **hidden_layer_sizes is default** set to 100 which seemed tad-bit on the higher end. So, I tried tuning the parameter and surprisingly, I was able to achieve almost the same results with the number of hidden layers set to around 45 to 50.
The trade-off here is that, increasing the size may cause the model to process slowly because of the several computations.

I was able to notice one thing when it came to hidden layers. Trying my luck with larger hidden nodes caused the accuracy to rather drop quickly giving hints of overfitting because of previous state knowledge.

6. Additional Work

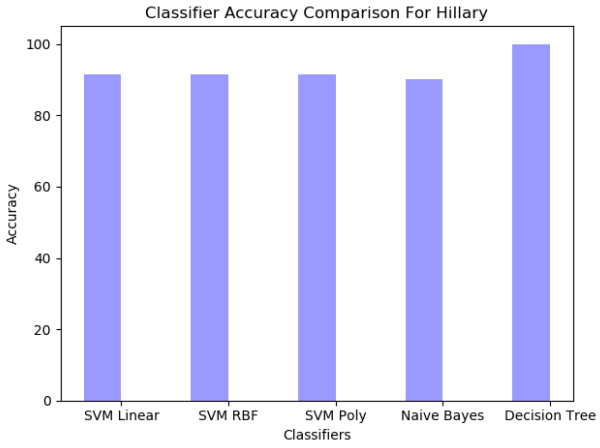
I wanted more perspetive on these models. Hence, I tried my hand at simple sentiment analysis model that uses Decision tree, naïve bayes and SVM. The dataset filled with hashtags from twitter were collected from Harvard Dataverse. It had tweets relating to 2016 US Elections on Hilary Clinton and Donald Trump.

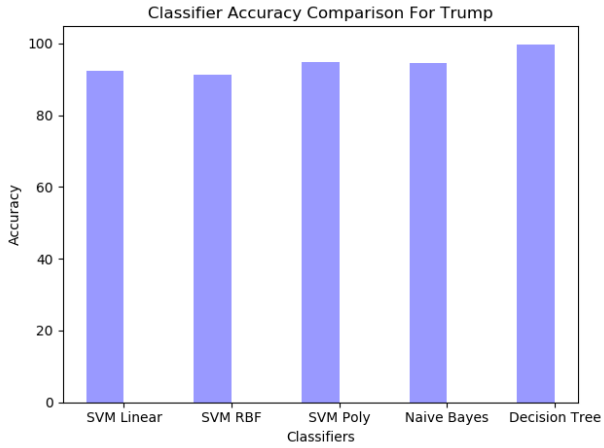
https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/PDI7IN&version=3.0

Once the Data was acquired, Labelling and removal of stop words were implemented as pre-processing.

Later, the extracted strings of data were compared with a bag of words containing positive, negative and neutral sentiments.

The classification of the tweets were done in all the three algorithms mentioned and compared.





In contrast to what was obtained with the DS-1 and DS-2, here, decision trees seem to provide the **highest** accuracy.

This may very well be because of the less number of predictors as well as a properly pre-processed dataset.

CONCLUSION

It is fair to say, based on the observations made above, that it is not possible for one algorithm/model to perform excellent under every scenario: dataset. This is because, every model has it's own perks and trade-off that may or may not suit a few datasets out there.

The sentiment analysis is one such example that proves that it all boils down to what the dataset is made of and what the user expects out of the model

That being said, there were times where I had to stop and think what exactly is going on.

- The time when decision tree gave an accuracy of 28%
- When SVM produced a result of 94.5% AFTER tuning.
- Neural Network's results before tuning.

If I were to continue with this project, there are a few topics I would like to definitely investigate,

- The hyperplanes formed by the SCM classifier.
- The fact that increase in the number of hidden layers causes a downfall in the accuracy of the ANN model.
- Tuning of hyper parameters in a more sophisticated and large scale manner.
- Try modelling on more datasets with larger samples.

Unasnwered Questions:

- Can I get better results from the other set of Naïve bayes classifiers?
- Why is decision tree still popular when we have Random Forest**?**
- Is neural Networks superior to all the models**?**

What now?

This to me was a really interesting and demanding mini-project. I really feel like I learned a lot from doing this. That being said, a lot could have been done if time wasn't exactly a constraint. I think that since there's space for more discovery and betterment, I will end up testing and trying out new ideas to improve on this.

REFERENCES:

- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- https://www.kaggle.com/viznrvn/optimal-parameters-for-svc-using-gridsearch
- https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769
- https://medium.com/@aneesha/svm-parameter-tuning-in-scikit-learn-using-gridsearchcv-2413c02125a0
- https://www.kaggle.com/hhllcks/neural-net-with-gridsearch
- https://www.youtube.com/watch?v=Gol_qOgRqfA
- https://blender.stackexchange.com/questions/84980/print-a-timer-in-python-to-see-how-long-to-run-a-script
- https://community.alteryx.com/t5/Data-Science-Blog/Why-use-SVM/ba-p/138440
- https://data-flair.training/blogs/svm-kernel-functions/
- http://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/
- https://www.quora.com/Why-did-SVM-become-more-popular-than-Logistic-Regression-and-what-are-the-implications-in-terms-of-using-the-Convex-Optimization-approach
- https://chrisalbon.com/machine_learning/support_vector_machines/svc_parameters_using_rbf_kernel/