ASSIGNMENT 2

Machine Learning

Introduction:

In this assignment, we are working on Support Vector Machines, Decision trees and Boosting. We have used two data sets:

1) Brisbane weather predictor
2) Best GPU Processor Predictor

DATASET 1(*Excel file attached with the Assignment).*

Brisbane weather dataset contains 24 variables. In this our target variable is RainTomorrow. It contains 56240 rows. We have done data pre processing as follows. I have used this dataset because weather is one of the most "wanted to know" everyday parameter. If we are able to predict weather forecast for upcoming days, it will be really beneficial for the market. As many business and our day to day activities are dependent on weather.

To better understand this data , I have done preprocessing of the data,so that it becomes easy for me to use it for various algorithms. As our target variable is RainTomorrow. It is a binary column with the values "Yes " and "No" depicting whether it will rain or not tomorrow.

1) First of all , we will check whether the data has any null values

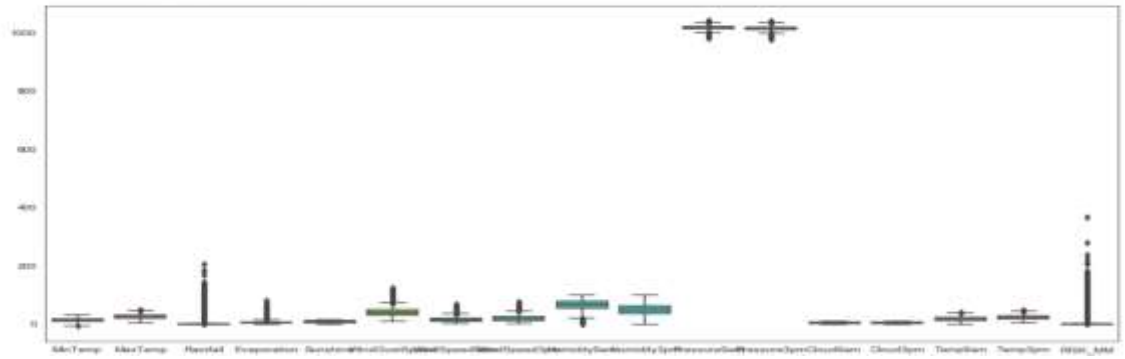
```
In [95]: #checking for NA
         newdf1.isnull().sum()

Out[95]: Date             0
         Location         0
         MinTemp          0
         MaxTemp          0
         Rainfall         0
         Evaporation      0
         Sunshine         0
         WindGustDir      0
         WindGustSpeed    0
         WindDir9am       0
         WindDir3pm       0
         WindSpeed9am     0
         WindSpeed3pm     0
         Humidity9am      0
         Humidity3pm      0
         Pressure9am      0
         Pressure3pm      0
         Cloud9am         0
         Cloud3pm         0
         Temp9am          0
         Temp3pm          0
         RainToday        0
         RISK_MM          0
         RainTomorrow     0
         dtype: int64
```

We can see that there are no null values present in the dataset.

2) We will check whether the data set contains any outliers or not:

```
plt.figure(figsize=(16,8))
sns.boxplot(data = newdf1 )
```

```
Out[96]: <matplotlib.axes._subplots.AxesSubplot at 0x1b5a7b2a8c8>
```



We can see there are no outliers present. There are some outliers in the RISK_MM Variable. As per the data description, it is given that RISK_MM has to be dropped off.

3) Exploring categorical variable:
There are 7 variables:
There is a date variable. It is denoted by Date column. There are 6 categorical variables. These are given by Location, WindGustDir, WindDir9am, WindDir3pm, RainToday and RainTomorrow. There are two binary categorical variables - RainToday and RainTomorrow. RainTomorrow is the target variable.

As we have these many categorical variables, we will be taking dummy variables by get dummy function. After this our data set has 91 columns and 56240 entries.

```
In [141]: newdf2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56420 entries, 0 to 56419
Data columns (total 91 columns):
MinTemp                    56420 non-null float64
MaxTemp                    56420 non-null float64
Rainfall                   56420 non-null float64
Evaporation                56420 non-null float64
Sunshine                   56420 non-null float64
WindGustSpeed              56420 non-null int64
WindSpeed9am               56420 non-null int64
WindSpeed3pm               56420 non-null int64
Humidity9am                56420 non-null int64
Humidity3pm                56420 non-null int64
Pressure9am                56420 non-null float64
Pressure3pm                56420 non-null float64
Cloud9am                   56420 non-null int64
Cloud3pm                   56420 non-null int64
Temp9am                    56420 non-null float64
Temp3pm                    56420 non-null float64
Year                       56420 non-null int64
Month                      56420 non-null int64
Day                        56420 non-null int64
Location_Brisbane          56420 non-null uint8
Location_Cairns            56420 non-null uint8
Location_Canberra          56420 non-null uint8
```

4) Scaling the data:

```
In [149]: # scaling the data
          from sklearn.preprocessing import StandardScaler
          sc = StandardScaler()
          X1_train = sc.fit_transform(X1_train)
          X1_test = sc.transform(X1_test)
```

We have done our data preprocessing and have split our data set in 70:30 ratio for training and test respectively. Now we can apply support vector machine algorithm, decision tree and boosting algorithm.

Support vector Machines:

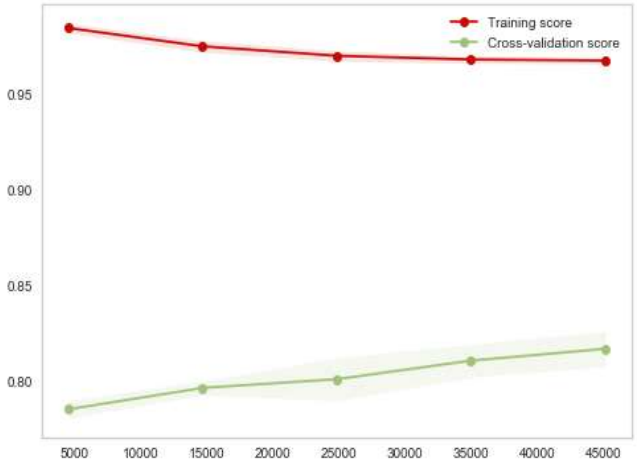
In machine learning, support-vector machines are supervised learning models which are used for classification and regression analysis. Here we can use different kernels to fit the data. We have used 3 types of SVMs. 1) Linear SVM 2) RBF SVM 3) Polynomial SBF and have measured accuracy rate for them.

K- Fold cross validation - In the basic approach, called k-fold CV, the training set is split into k smaller sets.The following procedure is followed for each of the k "folds":A model is trained using k-1 of the folds as training data. The resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. We have applied k-fold cross validation and have computed the average accuracy for all the three kernels.

Models	Accuracy Score	Accuracy mean after K-fold
Linear	85.9269762495569	85.59528147701398
RBF	86.44097837646225	86.2384159266244
Polynomial	77.36695112490303	85.54716514532309

As we can see that after k-fold our mean accuracy is highest for RBF as compared to linear and polynomial.

As we can see here that we are getting the maximum accuracy for RBF , kernel. We will plot the learning curve for the same. Learning Curve is a graph which shows model performance. A function has been defined to plot the learning curve in the code file. It plots the accuracy of training data and cross validation sets.



Explanation:

The learning curve shows that the training score is coming down as the training size increases and also shows that testing or the CV score is slowly increasing with the increase in Training size.

Decision Trees:

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The

tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

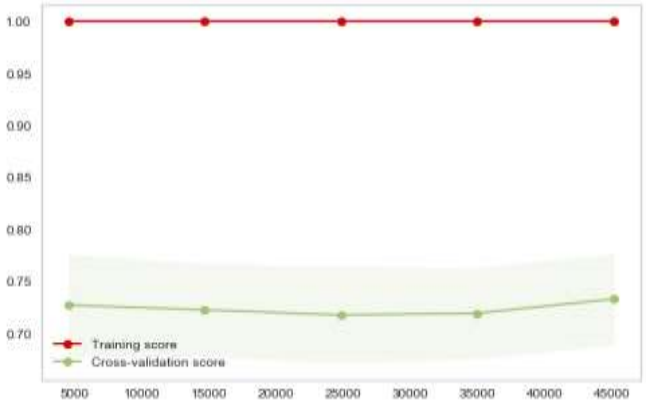
Accuracy chart for Decision tree is as follows:

Model	Accuracy score	Classification Report accuracy	Mean of accuracy after k-fold
gini	80.17842372681082	80	79.81713978188235

From the above, the accuracy for gini is computed as 80 percent.

We have used gini instead of entropy because entropy is slightly slow to compute. In both the algorithm our main motive is to find the best feature to classify our data better.

Learning curve for above model is:



Explanation:

The error rate is stuck at 1 for all training set sizes and cross validation error increases in U shape

Pruning:

It is a technique in machine learning and search algorithms that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.
After Pruning our model accuracy increased to 84 percent from 80 percent. At first, GridsearchCV is used to identify the best parameters for pruning which can give the best results. On running, we get the following output with the best parameters, max depth and minimum number of sample leaf:

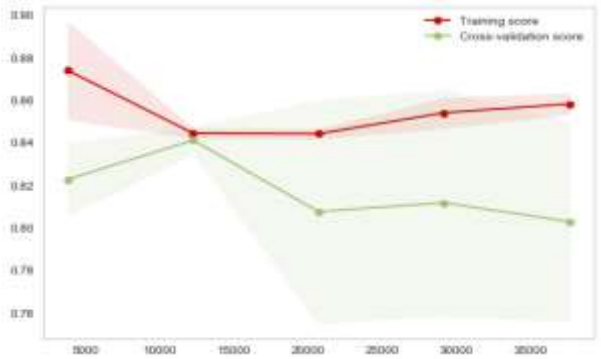
```
0.8462804476629362
{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 100}
```

Boosting:

The term ‘Boosting’ refers to a family of algorithms which converts weak learner to strong learners. To find weak rule, we apply base learning (ML) algorithms with a different distribution. Each time base learning algorithm is applied, it generates a new weak prediction rule. This is an iterative process. After many iterations, the boosting algorithm combines these weak rules into a single strong prediction rule.
After applying K-fold cross validation , our accuracy chart for boosting is as follows:

Model	Accuracy score	Classification report accuarcy	Accuracy mean after K-fold
Gradient Boost	83.25652841781874	83	83.1999
AdAboost	85.40115798180314	86	85.06606171669043

Here we can see that we have higher accuracy for Adaboost. The learning curve for the same is :



Explanation:

The train error and the test error converge at around 13000 Training size and slowly increases as the training size is increased.

After applying pruning to boosting algorithm, accuracy increased to 85.69402947283131

DATASET 2:

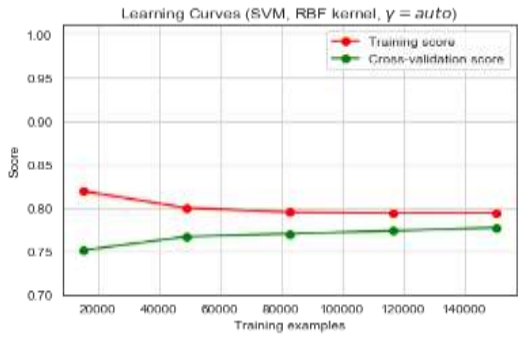
The second data set is aimed to find the ideal combination of processors to get the maximum GPU runtime. We have done the data pre processing same as in the previous code of logistic regression.

Support Vector Machine

Part1: Starting with Support vector machines , in this dataset as well, we have used three kernels. 1) linear 2) RBF 3) Polynomial. The accuracy chart for the three kernels is given below:

Model	Accuracy Score	Classification Report Accuracy	Average accuracy after K fold
Linear	75.6865787432118	76	75.70669625400848
RBF	78.7323506594259	79	78.55342600001222
Polynomial	77.36695112490303	77	77.16430583411518

Here we can see that the maximum accuracy is for RBF kernel. Hence the learning curve for RBF will be:



Explanation:

Here we can see that the training score slowly decrease as the training set size is increased and validation score is increasing with the increase in data size.

Decision Trees:

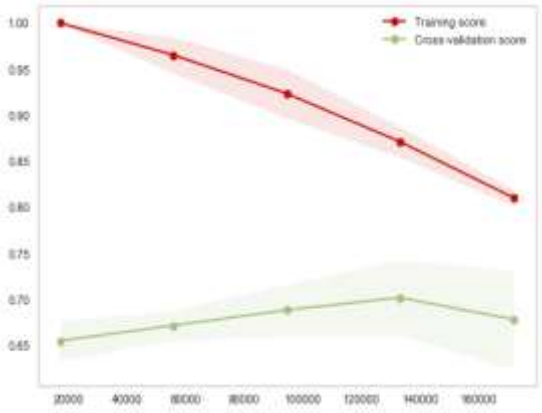
Experimenting with Decision trees on dataset 1, I came across the following figures for our model:

Model	Accuracy score	Classification Report accuracy	Mean of accuracy after k-fold
gini	68.2932505818464	68	68.54631086806523

From the above, the accuracy for gini is computed as 68 percent.

We have used gini instead of entropy because entropy is slightly slow to compute. In both the algorithm our main motive is to find the best feature to classify our data better.

The learning curve for the above model is:



Explanation:

The training error decreases with the increase in training set size and validation score first increases and then decreases at 13000. Data set needs to be smaller for cross validation to analyze it fully.

After Pruning our model accuracy increased to 78 percent from 68 percent. At first, GridsearchCV is used to identify the best parameters for pruning which can give the best results. On running, we get the following output with the best parameters, max depth and minimum number of sample leaf:

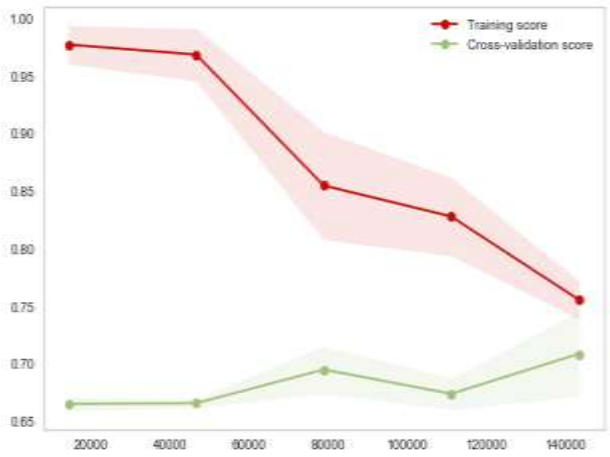
```
0.7871168948617863
{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 5}
```

Boosting:

I have used both Gradient descent algorithm and Gradient Descent algorithm for Boosting.

Model	Accuracy score	Classification Score	Mean accuracy after cross validation
Adaboost	75.45849495733127	75	75.35359353529385
GradientBoost	71.89604344453065	72	72.06266866675838

Here we can see that we are getting better accuracy for Adaboost which is 75 percent. We have drawn the learning curve for the same which is as follows:



Both AdaBoost and Gradient Boosting build weak learners in a sequential fashion. Originally, AdaBoost was designed in such a way that at every step the sample distribution was adapted to put more weight on misclassified samples and less weight on correctly classified samples. The final prediction is a weighted average of all the weak learners, where more weight is placed on stronger learners.

I have done pruning here for the best boosting tree. I have done Grid Search to find the best parameters for the boosting and find the best model. My Accuracy after pruning here is 0.7531569392817007. Accuracy after pruning may increase or decrease depending whether our model was overfitting or underfitting.

CONCLUSION:

After applying the three algorithms, we can see that highest accuracy we are getting for SVM with RBF kernel. Hence that is our best classification model. We have applied cross validation to improve our validation score and improve our training model to get better test accuracies. Pruning was done in both decision tree and boosting as it has reduced the complexity of our model and ultimately tweak our model to get the best results on testing set.
The additional things that we could have done are :

1) Adding more training instances would have given better svm accuracy.
2) Reducing the numbers of features in the training data we currently use. The algorithm will still fit the training data very well, but due to the decreased number of features, it will build less complex models. This should increase the bias and decrease the variance.
3) Hyperparameter optimization