PROBLEM STATEMENT :

Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied.

Let us first summarize the past observations from PHASE 2 and PHASE 3 before proceeding with PHASE 4 documentation :

PHASE 2 and 3 documentation summary :

1) A strong positive correlation exists between the amount of Sales and Customers of a store.

2) A positive correlation exists between the stores that had a Promo equal to 1(stores running promotions) and number of Customers.

3) For observations with Promo2 equal to 1 (that is for consecutive promotion), the number of Customers and Sales has a negative correlation.

4) Sales and Number of customers peak around Christmas timeframe.

5) Minimum number of customers are generally around the 24th of the month.

6) Most customers and Sales are around 30th and 1st of the month.

7) Minimum number of customers are generally on Saturday.

8) Most customers and Sales are on Sunday and Monday.

9) It is very important to properly process/map the features 'StateHoliday' and 'PromoInterval' before feeding the data to ML algorithms. Before mapping these features, the model scores were all over the place and only after mapping these features, I was able to get reasonable model performances.

10) Tree algorithms won’t even work if we get the dummy variables for 'StateHoliday', 'Assortment' and 'StoreType' features. Only after label encoding these features, the decision algorithm started to work/give scores.

11) After performing grid search hyperparameter tuning on decision tree algorithm, we find that the best decision tree model has max_depth of 2 and min_samples_leaf value of 5.

12) The best Decision tree model has a test R2 score of 94.79%, test MAPE of 6.83% and a test RMSPE of 9.49%.

13) From the Error distribution plot, we observe that there are few stores with higher error. But majority of the other points appear to be in a small horizontal area around zero which indicates that variance in the error is small and centered around zero.

PHASE 4 Documentation :

Ensemble Techniques:

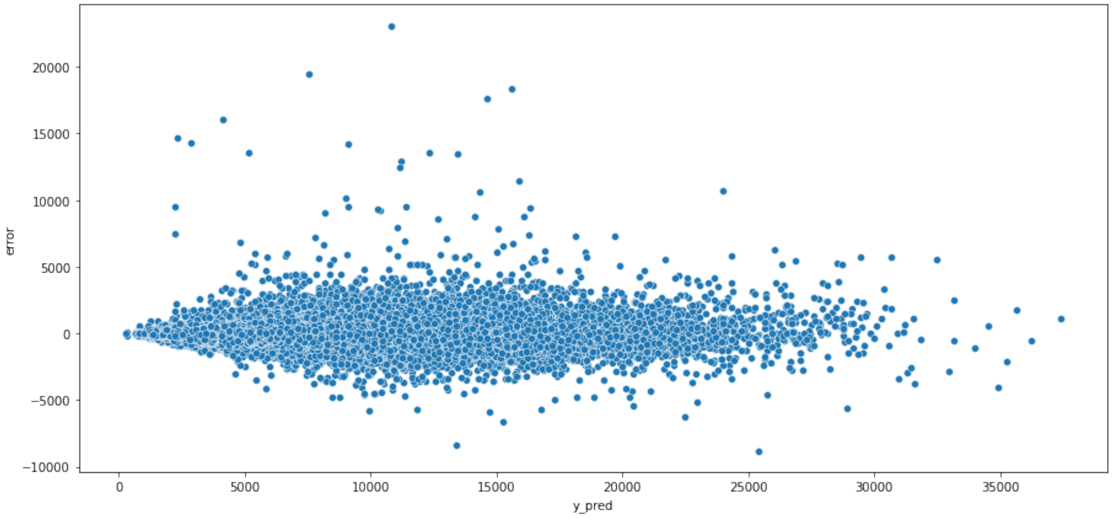
Random Forest :

Random Forest is an ensemble ML model that uses multiple decision trees through a technique called Bootstrapping and Aggregation known as bagging. Random Forest combines multiple decision trees and arrives at final output rather than relying on individual decision trees. Random Forest has multiple decision trees as base learning models and rows, and features are randomly sampled from the dataset forming sample datasets for every model.

Random Forest Model Train Score : 0.9999995196333464 , Random Forest Model Test Score : 0.9481023206156011
Train MAE : 122.3835 , Test MAE : 329.9142
Train MAPE : 0.0181 , Test MAPE : 0.0487
Train RMSE : 186.7553 , Test RMSE : 499.4189
Training RMSPE : 2.5055 , Testing RMSPE : 6.6389

We can observe that the Random Forest model performance is better than that of Decision Tree model.

Error Distribution of Random Forest Model :



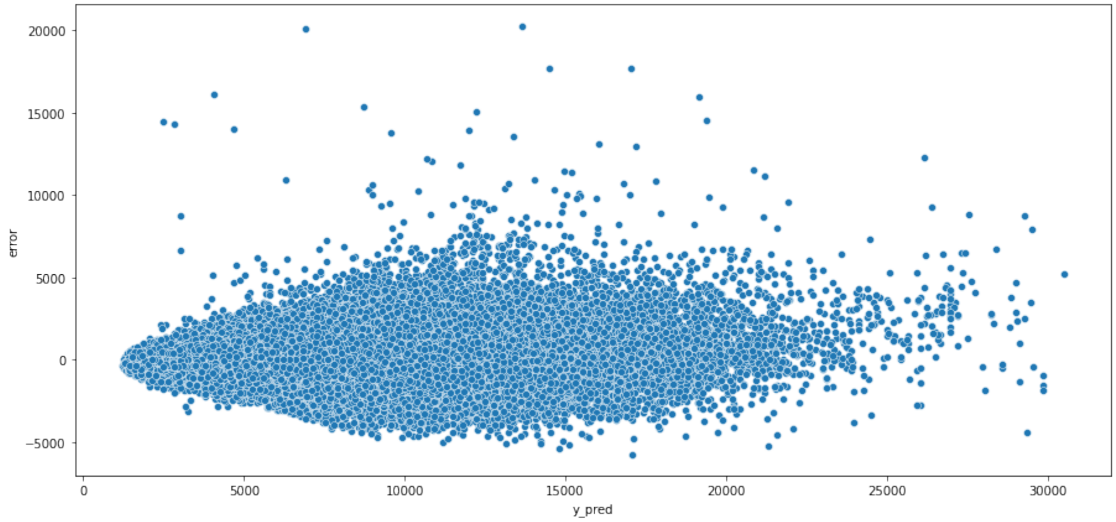
From the above plot, we can observe that the error distribution is much narrower in Random Forest model than in Decision Tree model and there are few stores with higher error. The variance in the error is smaller than Decision Tree model and centered around zero.

XGBoost :

Gradient boosting refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modeling problems. Ensembles are constructed from decision tree models. Trees are added one at a time to the ensemble and fit to correct the prediction errors made by prior models. This is a type of ensemble machine learning model referred to as boosting. XGBoost expects to have the base learners which are uniformly bad at the remainder so that when all the predictions are combined, bad predictions cancel out and better one sums up to form final good predictions.

XGBoost Model Train Score : 0.8849876705183609 , XBGoost Model Test Score : 0.8854866955603585
Train MAE : 771.3302 , Test MAE : 771.8178
Train MAPE : 0.1173 , Test MAPE : 0.1173
Train RMSE : 1051.5082 , Test RMSE : 1052.8874
Training RMSPE : 16.6002 , Testing RMSPE : 16.3152

Error Distribution of XGBoost Model :



From the above plot, we can observe that the error distribution of XGBoost model is much wider than Random Forest and Decision Tree models and there are comparitively many stores with higher error. The variance in the error is wider than Decision Tree and Random Forest models but centered around zero.

Deep Learning MLP Model :

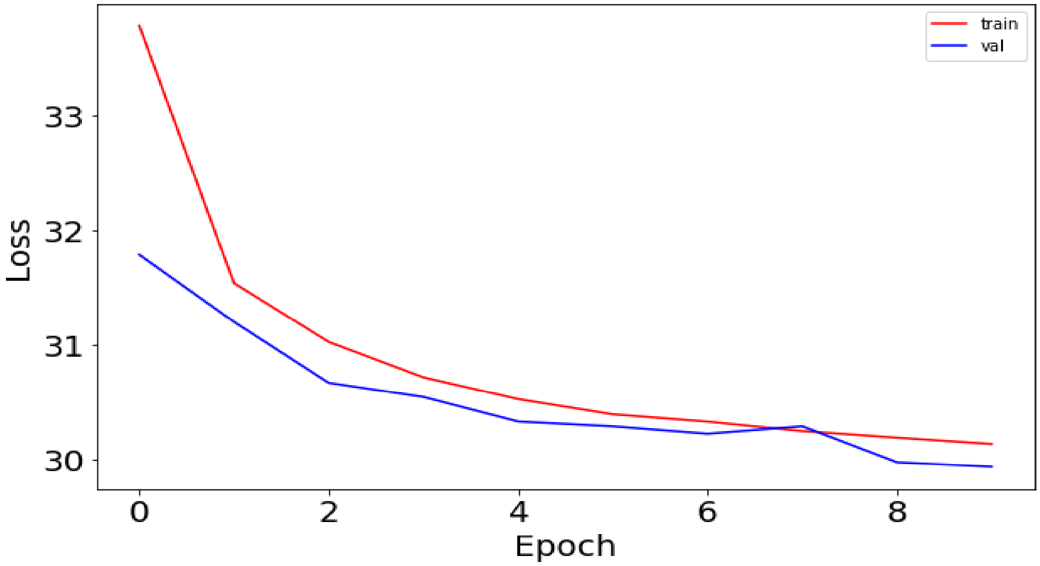
MLP is a neural network for regression/classification tasks.The data flows in a single direction, that is forward, from the input layers to hidden layer(s) and than to output layer. Backpropagation is a technique where the multi-layer perceptron receives feedback on the error in its results and the MLP adjusts its weights accordingly to make more accurate predictions in the future.

A simple MLP model has a single hidden layer of nodes with an output layer used to make a prediction.

Here, we need to emphasize on the shape of the input dimension which is equal to the number of features in x train data.

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)                  (None, 50)                900

 dense_1 (Dense)                  (None, 1)                  51

=================================================================
Total params: 951
Trainable params: 951
Non-trainable params: 0
```

Let us plot the loss function as a function of epochs for the training and test sets to see how the network performed as follows :



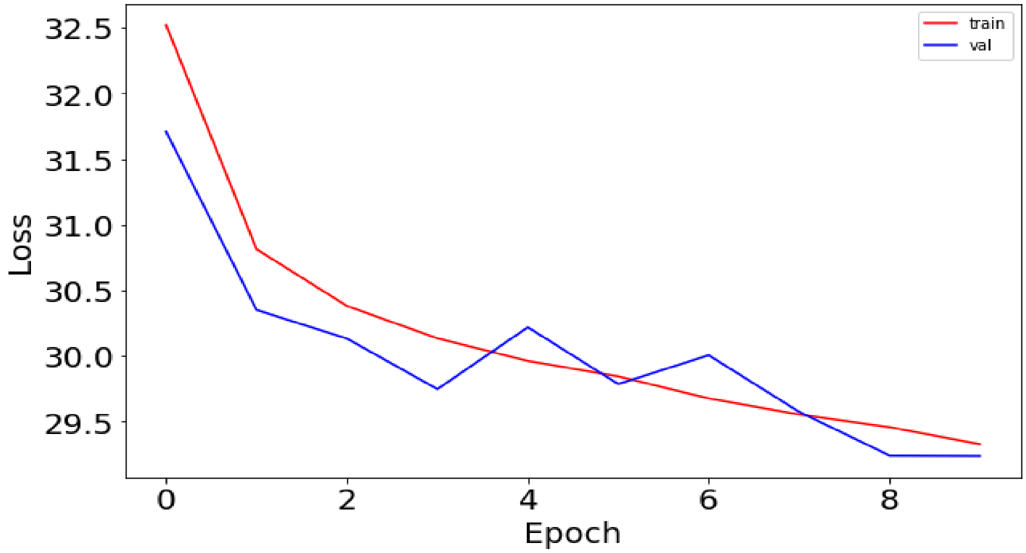
Let us evaluate the model performance against test data :

test_MLP_score : 0.7835039934276156
Test Data loss : 896.51
Test Data mean_absolute_error : 896.51

Now, let us increase the number of neurons and see if it changes the model performance :

```
 Layer (type)              Output Shape              Param #
=================================================================
 dense_2 (Dense)              (None, 350)               6300

 dense_3 (Dense)              (None, 1)               351

=================================================================
Total params: 6,651
Trainable params: 6,651
Non-trainable params: 0
_________________________________________________________________
```

Let us plot the loss function as a function of epochs for the training and test sets to see how the network performed as follows :



Let us evaluate the model performance against test data :

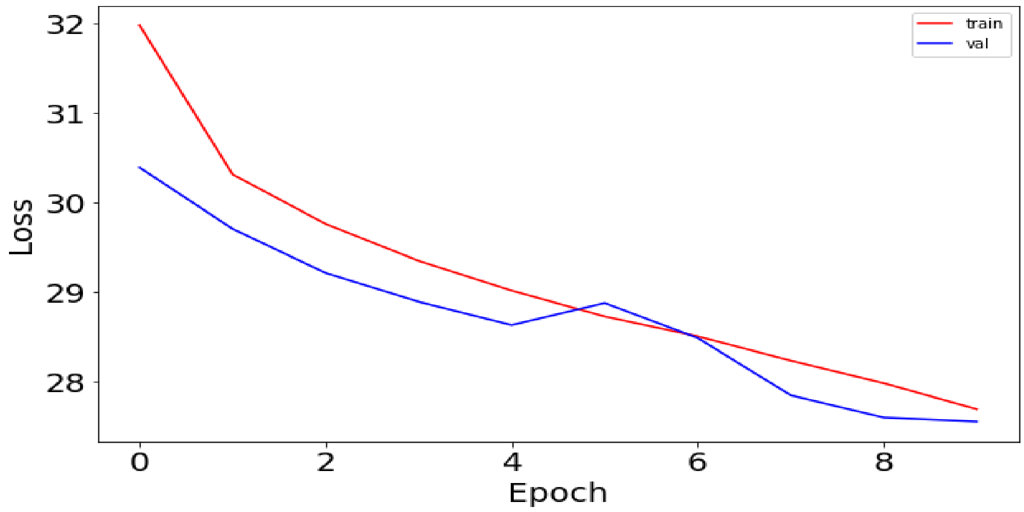
test_MLP_score : 0.8090183403205422
Test Data loss : 853.54
Test Data mean_absolute_error : 853.54

Now, let us create a MLP model with additional set of layers and keeping the number of neurons as 300 as that gave a better model performance earlier :

```
_________________________________________________________________
 Layer (type)              Output Shape              Param #
=================================================================
 dense_4 (Dense)              (None, 350)               6300

 dense_5 (Dense)              (None, 350)               122850

 dense_6 (Dense)              (None, 350)               122850
```

```
 dense_7 (Dense)           (None, 350)            122850

 dense_8 (Dense)           (None, 1)             351

=================================================================
Total params: 375,201
Trainable params: 375,201
Non-trainable params: 0
_________________________________________________________________
```

Let us plot the loss function as a function of epochs for the training and test sets to see how the network performed as follows :



Let us evaluate the model performance against test data :

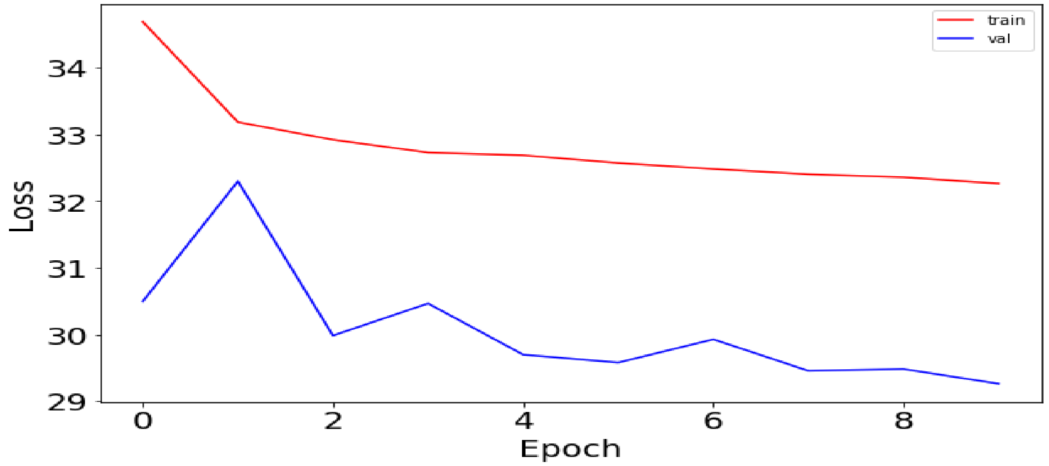
test_MLP_score : 0.8546873907288122
Test Data loss : 757.39
Test Data mean_absolute_error : 757.39

When the networks grow deeper, there is a chance of too much of learning from the training data and overfit to it.Dropout is a simple and powerful way to prevent overfitting. Some neurons will be randomly selected and dropped from the network in each layer.Dropout layer needs to be added after activation layer.

```
_____________________________________________________________
 Layer (type)              Output Shape           Param #
=================================================================
 dense_9 (Dense)           (None, 350)            6300

 dropout (Dropout)        (None, 350)            0

 dense_10 (Dense)          (None, 350)            122850
```

```
dropout_1 (Dropout)       (None, 350)          0

dense_11 (Dense)          (None, 350)          122850

dropout_2 (Dropout)       (None, 350)          0

dense_12 (Dense)          (None, 350)          122850

dropout_3 (Dropout)       (None, 350)          0

dense_13 (Dense)          (None, 1)            351

=================================================================
Total params: 375,201
Trainable params: 375,201
Non-trainable params: 0
```

Let us plot the loss function as a function of epochs for the training and test sets to see how the network performed as follows :



Let us evaluate the model performance against test data :

test_MLP_score : 0.8293292282936097
Test Data loss : 856.74
Test Data mean_absolute_error : 856.74

Observations from Phase 4:

1) For our current case study, We observe that the Random Forest model performance is better than that of Decision Tree model.

2) The error distribution of Random Forest model is much narrower than the error distribution of Decision Tree model and there are few stores with higher error.

The variance in the error is smaller than Decision Tree model and centered around zero.

3) The best Random Forest model has a test R2 score of 94.81% , test MAPE of 4.87% and a test RMSPE of 6.63%.

4) The error distribution of XGBoost model is much wider than Random Forest and Decision Tree models and there are comparitively many stores with higher error. The variance in the error is wider than Decision Tree and Random Forest models, but centered around zero.

5) A basic MLP neural network with just a single hiden layer consisting of 50 neurons had a score of 78.35 % with MAE of approximately 896.

6) When we increased the number of neurons to 350 in this basic MLP network, the score increased to 80.9 % and MAE reduced to approximately 853.

7) When we increased the number of hidden layers to 4 with each hidden layer consisting of 350 neurons in the MLP network, the score increased to 85.4 % and MAE reduced to approximately 757.

8) When we introduced Dropout layer to this above MLP network to reduce overfitting , there was no noticeable improvement in model performance.

9) For our current project scenario, since we observe that the model performance of Decision Tree and Random Forest models are better then the MLP neural network,we will use the Decision tree or Random Forest models to deploy the code.

10) Given below are the comparison of all the models used in our current case study :

```
                                      Test R2 Score   Test MAE
Basic MLP with 50 Nuerons                  0.7835       896.51
Basic MLP with 350 Nuerons                 0.8090       853.54
MLP with 4 hidden layers                   0.8546       757.39
MLP with 4 hidden layers with Dropout        0.8293       856.74
```

```
             Train R2 Score  Test R2 Score  Train MAE  Test MAE  \
Linear Regression       0.826214       0.828777   938.1422  934.7298
Ridge Regression        0.825943       0.828526   939.6218  936.2103
Lasso Regression        0.825839       0.828425   940.0902  936.6873
Decision Tree         1.000000       0.948102     0.0084  463.4370
Random Forest           1.000000       0.948102   122.3835  329.9142
XGBoost Model           0.884988       0.885487   771.3302  771.8178
```

```
             Train MAPE  Test MAPE  Train RMSE  Test RMSE  Train RMSPE  \
Linear Regression      0.1429     0.1426   1292.5497  1287.4622      19.1589
Ridge Regression       0.1432     0.1429   1293.5579  1288.4075      19.1795
Lasso Regression       0.1432     0.1429   1293.9468  1288.7855      19.1878
```

Decision Tree	**0.0000**	**0.0682**	**2.1490**	**708.8069**	**0.0557**
Random Forest	**0.0181**	**0.0487**	**186.7553**	**499.4189**	**2.5055**
XGBoost Model	**0.1173**	**0.1173**	**1051.5082**	**1052.8874**	**16.6002**

	Test RMSPE
Linear Regression	**19.2301**
Ridge Regression	**19.2461**
Lasso Regression	**19.2537**
Decision Tree	**9.5074**
Random Forest	**6.6389**
XGBoost Model	**16.3152**

For our current project scenario, since we observe that the model performance of Decision Tree and Random Forest models are better than the MLP neural network, we will use the Decision tree or Random Forest models to deploy the code.