Milestone_UNO:

1. What do you think about classification Problem here? What principle will you follow to perform for your approach?

Answer: We know a classification problem is when the output variable is a category, such as "red" or "blue". And, a regression problem is when the output variable is a real value, such as "dollars" or "weight". On the other hand, a clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

In our dataset, we have got a data frame where six feature such as 'age', 'rok_index', 'rok_summation', 'promo_value' , 'event_value' , 'conversion_ratio', has been given and a class namely "class" as output variable also along with those feature. We can understand that our given problem is a classification problem. Here, the classification problem is the task of predicting a discrete class label such as 'a', 'b', or 'c' where a classification algorithm may predict a continuous value, but the continuous value is in the form of a probability for a class label.

Here, I have built a model pipeline where I used some basic approach such as data understanding through exploratory data analysis (EDA), build a data model using different classification algorithm such as Logistic Regression, KNearest (KNN), Support Vector Classifier, Decision Tree Classifier, Random Forest Classifier, Ada Boost Classifier, Gradient Boosting Classifier, and lastly Multilayer Perceptron (MLP). After creating our model, we evaluate our model by measuring accuracy for every classification algorithm that we have used. Then, we deploy our model on our test data and predict the class label for each distinct data along with their features.

2. Write some of the classification algorithms you might try on this TEST DATA SET. Explain the reason why would you use them?

Answer: In this classification problem, we have used Logistic Regression, KNearest (KNN), Support Vector Classifier, Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier, and Multilayer Perceptron (MLP) classification algorithm on test data.

1. **Logistic Regression:** This is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability. Here, we need to predict 'class' of our test data. Since, we want to predict something that's why to build a predictive model we use logistic regression.

2. **K-Nearest Neighbors (KNN)**: This is one of the simplest algorithms used in machine learning for classification problem. KNN algorithm uses data and classify new data points based on similarity measures by using distance function. Classification is done by a majority vote to its neighbors. The data is also assigned to the class which has the nearest neighbors. That's why we think this algorithm might be helped us to identify the correct class depends on their given features.

3. **Support Vector Classifier:** SVM is a supervised machine learning algorithm which can be used for classification problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs. Sometimes, it performs better instead of using other classification algorithm because of its kernel trick.

4. **Decision Tree Classifier:** Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. Where the goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. This process of learning simple decision to predict our class, we use this algorithm.

5. **Random Forest Classifier:** Random forests is a supervised learning algorithm. It can be used for classification. It is also the most flexible and easy to use algorithm to create decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting.

6. **Gradient Boosting Classifier:** we know gradient boosting is a greedy algorithm and can overfit a training dataset quickly. It can benefit from regularization methods such as L1, L2 regularization that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting. Since, we have got worst accuracy for previous classification algorithm then we used this algorithm to avoid overfitting and underfitting.

7. **Multilayer Perceptron**: MLP with one hidden layer are capable of approximating any continuous function. Multilayer perceptrons are often applied to supervised learning problems they train on a set of input-output pairs and learn to model the correlation between those inputs and outputs. To emphasize on the correlation between features and class label, we used this algorithm.

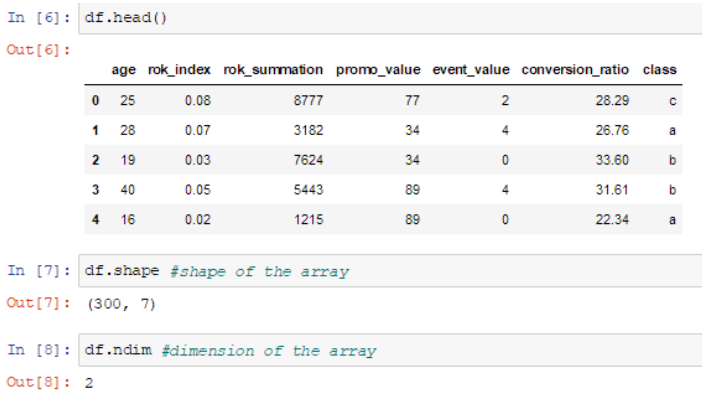
Milestone_DOS:

1. What do you think about **EDA?** How do you perform your EDA on a given dataset?

Answer: I think EDA helps to obtain specific knowledge specially check for missing data and other mistakes, gain maximum insight into the data set and its underlying structure, helps to build a model and explains the data with a minimum number of predictor variables, check assumptions associated with any model fitting or hypothesis test such as ANOVA test, null hypothesis using Z-test, t-test. As well as, it also useful for finding parameter estimates and their associated confidence intervals or margins of error and identify the most influential variables.

From given dataset, my exploratory data analysis (EDA) listed below:

1. Preview data:

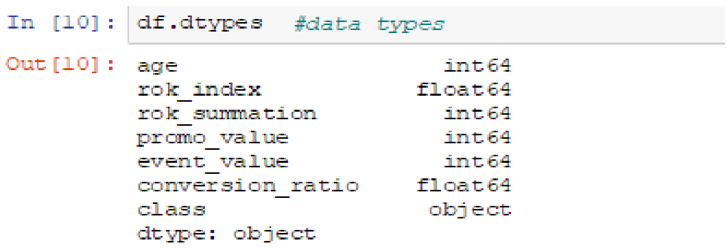


```
In [6]: df.head()
Out[6]:
```

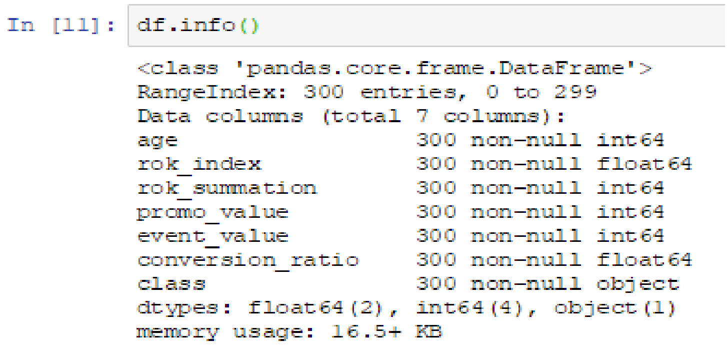
	age	rok_index	rok_summation	promo_value	event_value	conversion_ratio	class
0	25	0.08	8777	77	2	28.29	c
1	28	0.07	3182	34	4	26.76	a
2	19	0.03	7624	34	0	33.60	b
3	40	0.05	5443	89	4	31.61	b
4	16	0.02	1215	89	0	22.34	a

```
In [7]: df.shape #shape of the array
Out[7]: (300, 7)

In [8]: df.ndim #dimension of the array
Out[8]: 2
```

2. Check total number of entries and column types:

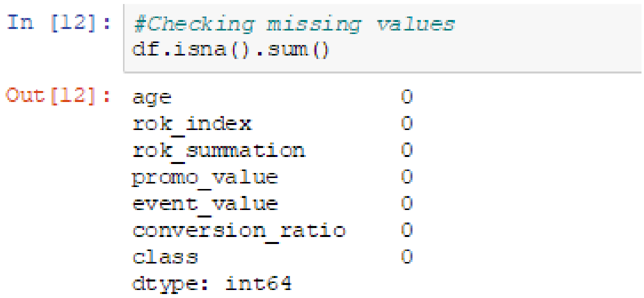


```
In [10]: df.dtypes   #data types

Out[10]: age                    int64
         rok_index            float64
         rok_summation          int64
         promo_value            int64
         event_value            int64
         conversion_ratio     float64
         class                 object
         dtype: object
```



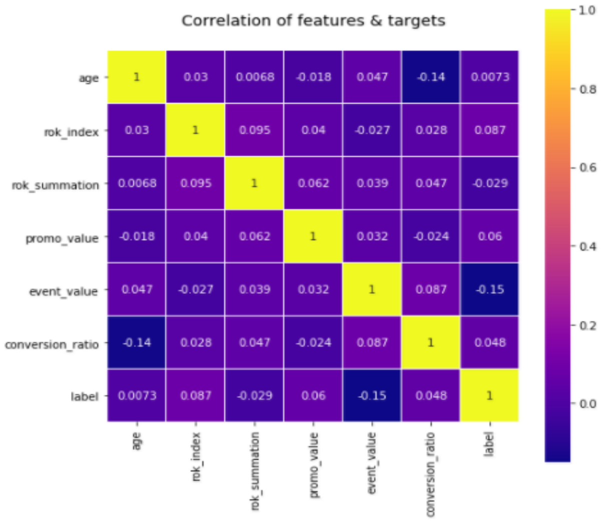
```
In [11]: df.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 300 entries, 0 to 299
         Data columns (total 7 columns):
         age                    300 non-null int64
         rok_index              300 non-null float64
         rok_summation          300 non-null int64
         promo_value            300 non-null int64
         event_value            300 non-null int64
         conversion_ratio       300 non-null float64
         class                  300 non-null object
         dtypes: float64(2), int64(4), object(1)
         memory usage: 16.5+ KB
```

3. Check any null values:

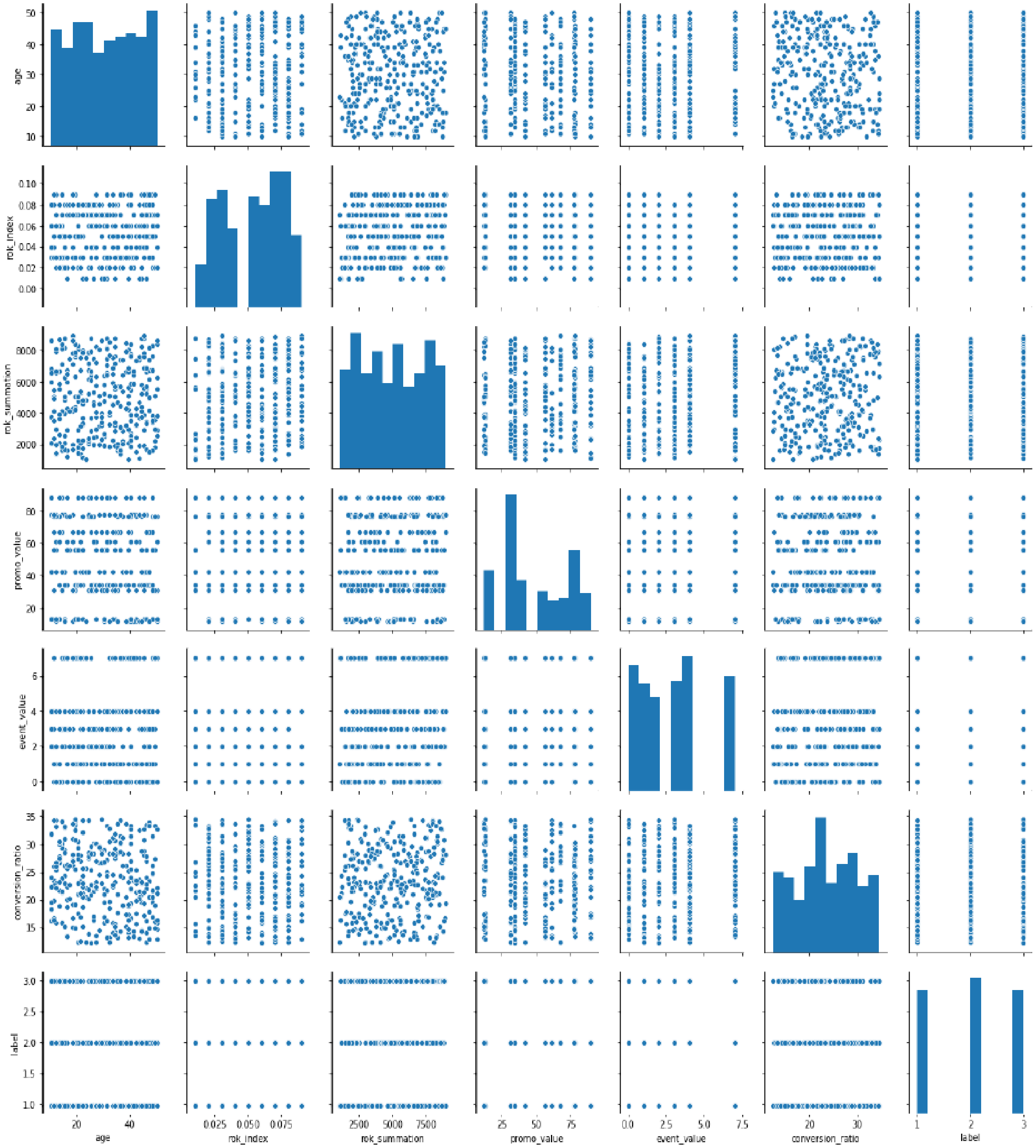


```
In [12]: #Checking missing values
         df.isna().sum()

Out[12]: age                   0
         rok_index             0
         rok_summation         0
         promo_value           0
         event_value           0
         conversion_ratio      0
         class                 0
         dtype: int64
```

4. Correlation between features and class visualization:

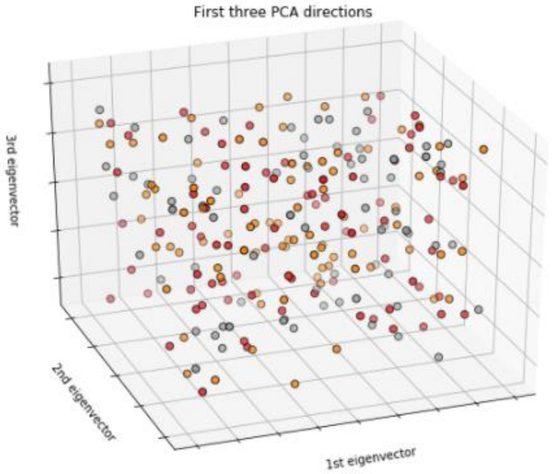


5. Plot distribution of numeric data (univariate and pairwise joint distribution):

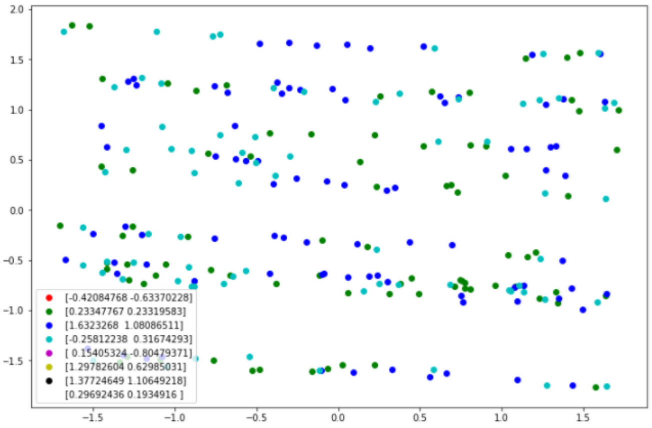


6. Univariate and multivariate graphical:

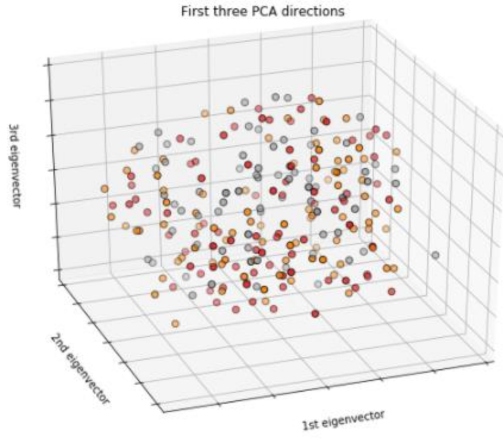
a. Principle Component analysis (PCA) 3D:



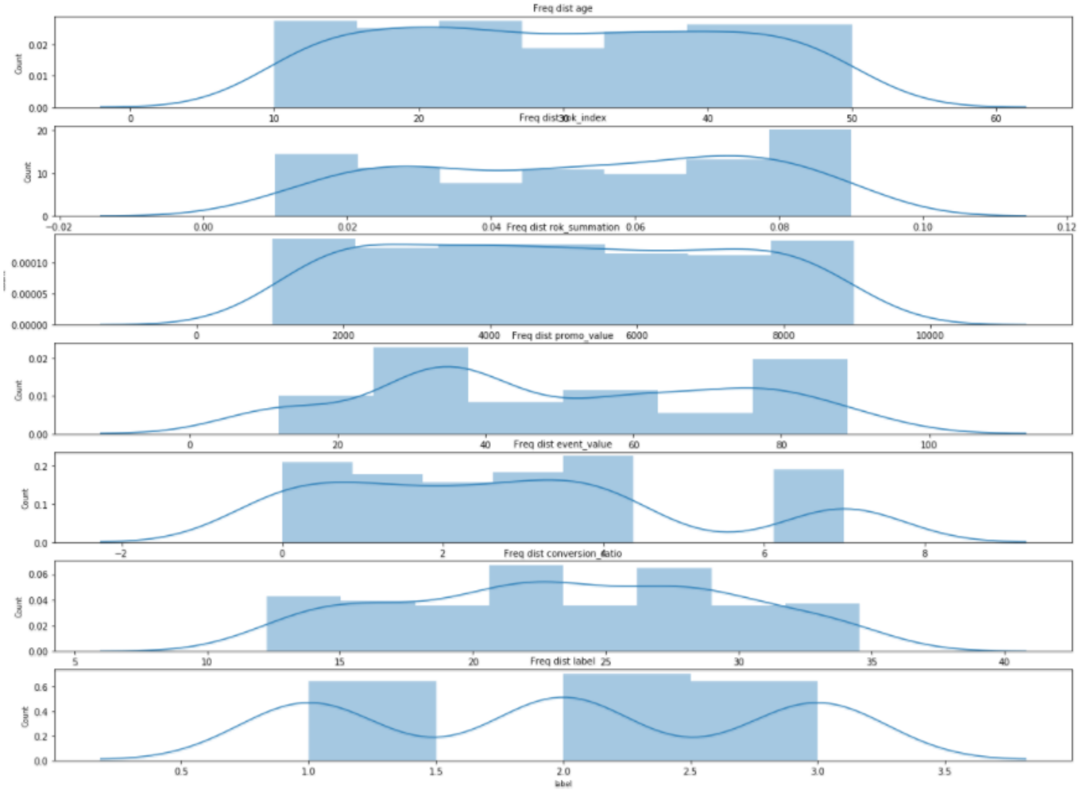
b. Principle Component analysis (PCA) 2D:



c. t-Distributed Stochastic Neighbor Embedding (t-SNE) 3D:

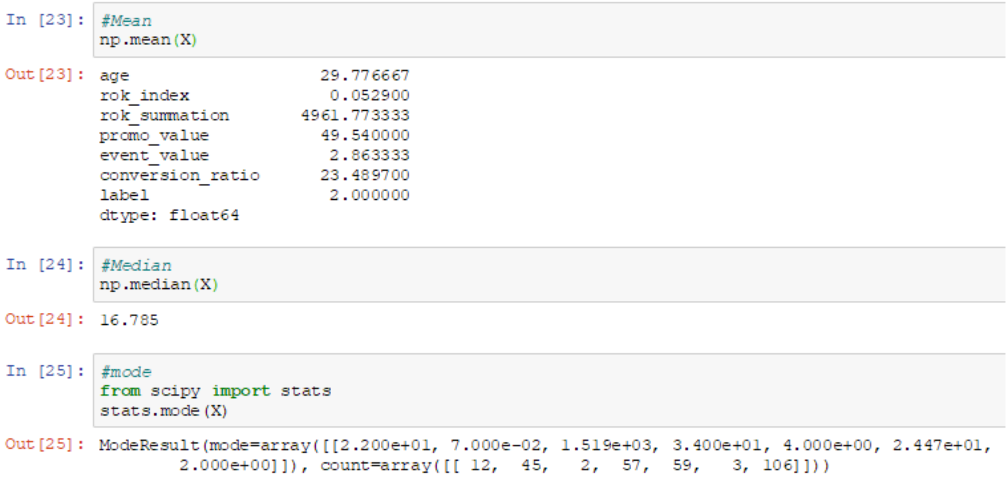


d. Univariate Distribution:



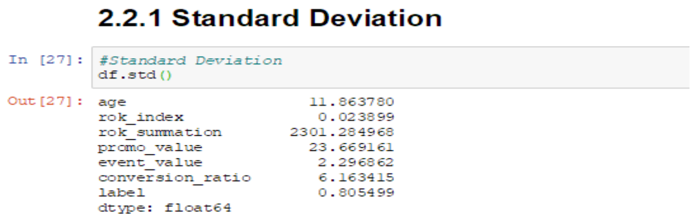
2. The **central** tendencies of my findings and fluctuations rate between data:

a. Mean, Median and Mode:



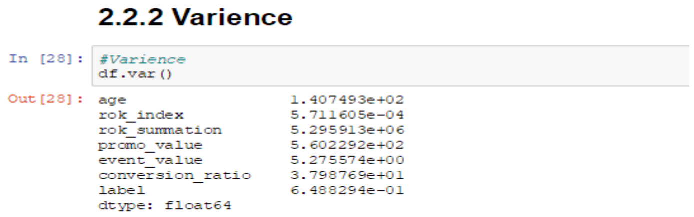
```
In [23]: #Mean
         np.mean(X)

Out[23]: age                     29.776667
         rok_index                0.052900
         rok_summation         4961.773333
         promo_value             49.540000
         event_value              2.863333
         conversion_ratio        23.489700
         label                    2.000000
         dtype: float64

In [24]: #Median
         np.median(X)

Out[24]: 16.785

In [25]: #mode
         from scipy import stats
         stats.mode(X)

Out[25]: ModeResult(mode=array([[2.200e+01, 7.000e-02, 1.519e+03, 3.400e+01, 4.000e+00, 2.447e+01,
                 2.000e+00]]), count=array([[ 12,  45,   2,  57,  59,   3, 106]]))
```

b. Standard deviation, variance and coefficient of variation:



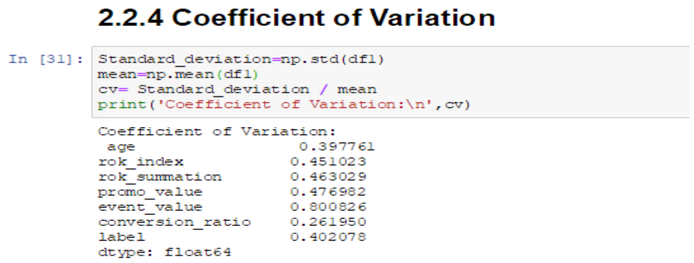
2.2.1 Standard Deviation

```
In [27]: #Standard Deviation
         df.std()

Out[27]: age                     11.863780
         rok_index                0.023899
         rok_summation         2301.284968
         promo_value             23.669161
         event_value              2.296862
         conversion_ratio         6.163415
         label                    0.805499
         dtype: float64
```



2.2.2 Varience

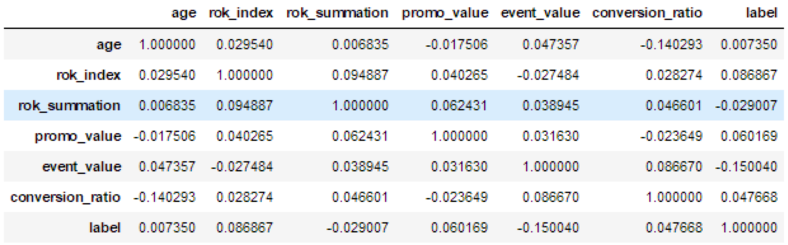
```
In [28]: #Varience
         df.var()

Out[28]: age                  1.407493e+02
         rok_index            5.711605e-04
         rok_summation        5.295913e+06
         promo_value          5.602292e+02
         event_value          5.275574e+00
         conversion_ratio     3.798769e+01
         label                6.488294e-01
         dtype: float64
```



2.2.4 Coefficient of Variation

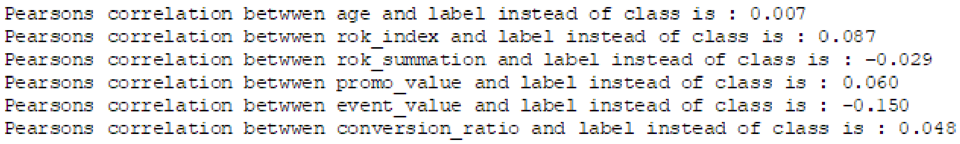
```
In [31]: Standard_deviation=np.std(df1)
         mean=np.mean(df1)
         cv= Standard_deviation / mean
         print('Coefficient of Variation:\n',cv)

         Coefficient of Variation:
          age                  0.397761
         rok_index            0.451023
         rok_summation        0.463029
         promo_value          0.476982
         event_value          0.800826
         conversion_ratio     0.261950
         label                0.402078
         dtype: float64
```

c. Correlation:



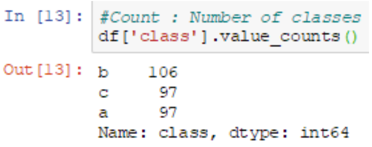
	age	rok_index	rok_summation	promo_value	event_value	conversion_ratio	label
age	1.000000	0.029540	0.006835	-0.017506	0.047357	-0.140293	0.007350
rok_index	0.029540	1.000000	0.094887	0.040265	-0.027484	0.028274	0.086867
rok_summation	0.006835	0.094887	1.000000	0.062431	0.038945	0.046601	-0.029007
promo_value	-0.017506	0.040265	0.062431	1.000000	0.031630	-0.023649	0.060169
event_value	0.047357	-0.027484	0.038945	0.031630	1.000000	0.086670	-0.150040
conversion_ratio	-0.140293	0.028274	0.046601	-0.023649	0.086670	1.000000	0.047668
label	0.007350	0.086867	-0.029007	0.060169	-0.150040	0.047668	1.000000

d. Pearson Correlation:



```
Pearsons correlation betwwen age and label instead of class is : 0.007
Pearsons correlation betwwen rok_index and label instead of class is : 0.087
Pearsons correlation betwwen rok_summation and label instead of class is : -0.029
Pearsons correlation betwwen promo_value and label instead of class is : 0.060
Pearsons correlation betwwen event_value and label instead of class is : -0.150
Pearsons correlation betwwen conversion_ratio and label instead of class is : 0.048
```

3. Which class does appear most in the "data.train" data set?



```
In [13]: #Count : Number of classes
         df['class'].value_counts()

Out[13]: b    106
         c     97
         a     97
         Name: class, dtype: int64
```

Here, we can see that class b appears mostly.

4. Any correlations with any other variables?

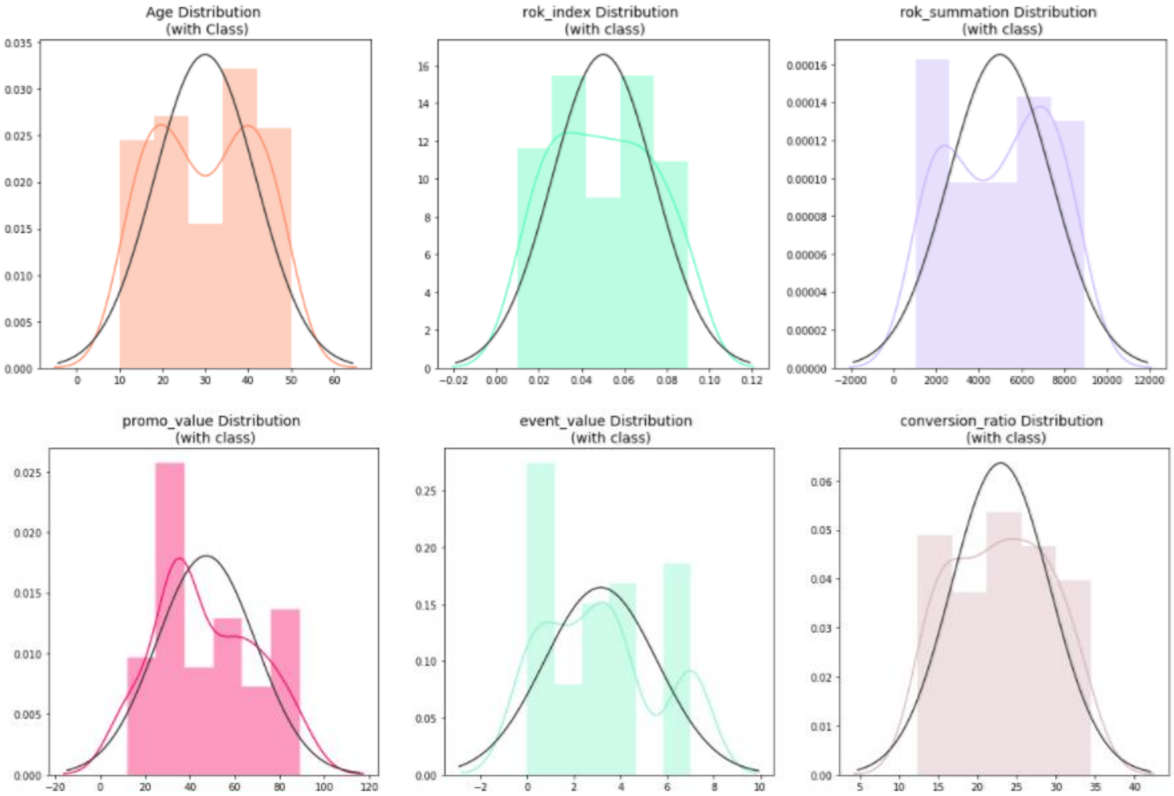
   Answer: We know, when correlation coefficient is close to +1 then there would be a large positive relationship and close to -1 then there would be a large negative relationship. On the other hand, when it's close to 0 then there is possibly no relationship.

   From correlation, we can see that rok_summation and event_value have no relationship with our class label. Also, rest of the features are also far from the positive relationship. So, we can conclude that there is very weak relationship between variables.

   We also know that if Pearson value is less than 0.001 then there is a strong certainty in the result, if Pearson value is less than 0.05 then there is a moderate certainty in the result, Pearson value is less than 0.1 then there is a weak certainty in the result and Pearson value is greater than 0.1 then there is a no certainty in the result. From getting values of Pearson, we can interpret that most of the variables don't have any certainty in the prediction result.

5. Are there any anomalies?

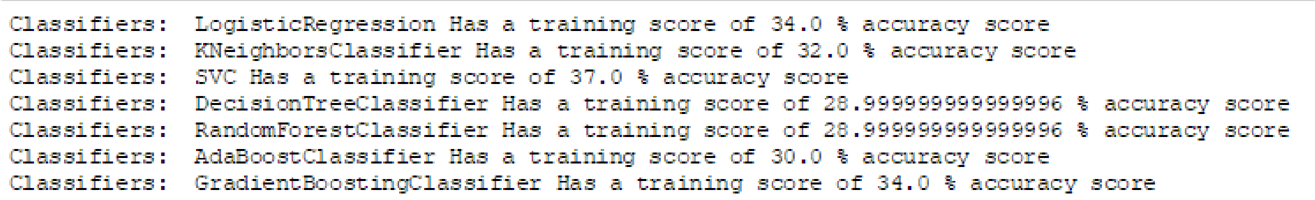
   Answer: We know that anomaly refers to the task of finding any observations that do not conform to the normal, expected behavior. And these observations can be named as anomalies.
   From PCA and t-SNE analysis, we can see that the plotting point values of class those depend on variables from train data are looking too noisy and it is difficult to predict class of test data. Below I have shown the anomalies distribution between class and other variables.



Milestone_TRES:

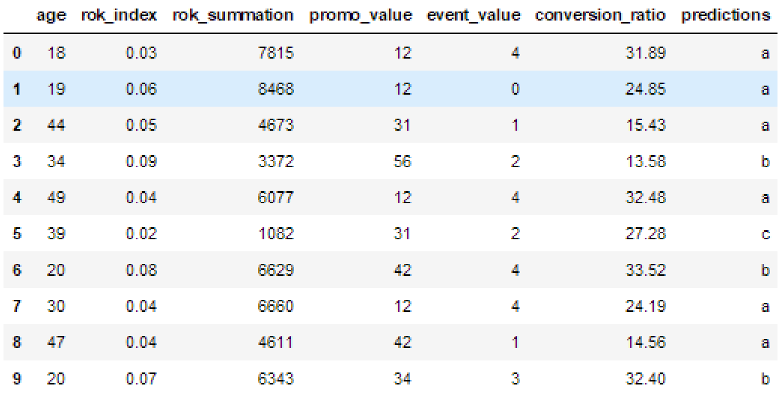
1. Model with Using any Library, ML Packages and Frameworks:

We have used different classification algorithm in our machine learning pipeline model and predict the “class” of “test.data”. Apart from the code, here I have attached model used classifier their performance and prediction label in test data based on model.



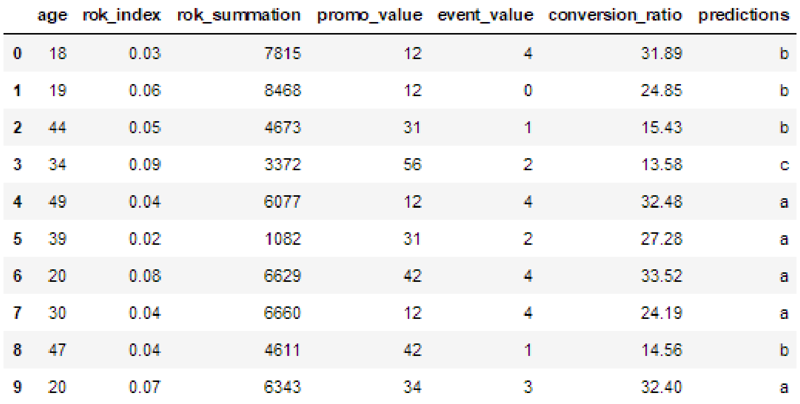
```
Classifiers:  LogisticRegression Has a training score of 34.0 % accuracy score
Classifiers:  KNeighborsClassifier Has a training score of 32.0 % accuracy score
Classifiers:  SVC Has a training score of 37.0 % accuracy score
Classifiers:  DecisionTreeClassifier Has a training score of 28.999999999999996 % accuracy score
Classifiers:  RandomForestClassifier Has a training score of 28.999999999999996 % accuracy score
Classifiers:  AdaBoostClassifier Has a training score of 30.0 % accuracy score
Classifiers:  GradientBoostingClassifier Has a training score of 34.0 % accuracy score
```

a. Prediction on test data using Logistic regression:



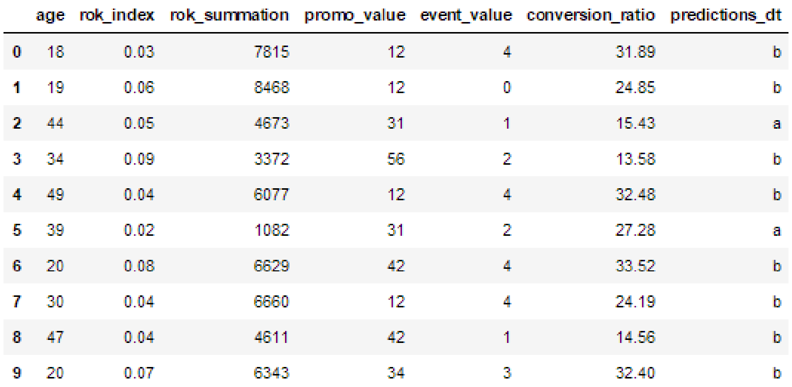
	age	rok_index	rok_summation	promo_value	event_value	conversion_ratio	predictions
0	18	0.03	7815	12	4	31.89	a
1	19	0.06	8468	12	0	24.85	a
2	44	0.05	4673	31	1	15.43	a
3	34	0.09	3372	56	2	13.58	b
4	49	0.04	6077	12	4	32.48	a
5	39	0.02	1082	31	2	27.28	c
6	20	0.08	6629	42	4	33.52	b
7	30	0.04	6660	12	4	24.19	a
8	47	0.04	4611	42	1	14.56	a
9	20	0.07	6343	34	3	32.40	b

b. Prediction on test data using KNN:



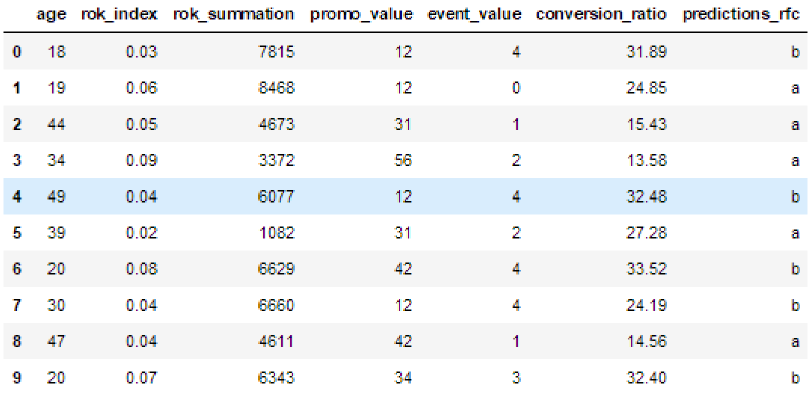
	age	rok_index	rok_summation	promo_value	event_value	conversion_ratio	predictions
0	18	0.03	7815	12	4	31.89	b
1	19	0.06	8468	12	0	24.85	b
2	44	0.05	4673	31	1	15.43	b
3	34	0.09	3372	56	2	13.58	c
4	49	0.04	6077	12	4	32.48	a
5	39	0.02	1082	31	2	27.28	a
6	20	0.08	6629	42	4	33.52	a
7	30	0.04	6660	12	4	24.19	a
8	47	0.04	4611	42	1	14.56	b
9	20	0.07	6343	34	3	32.40	a

c. Prediction on test data using Decision Tree:



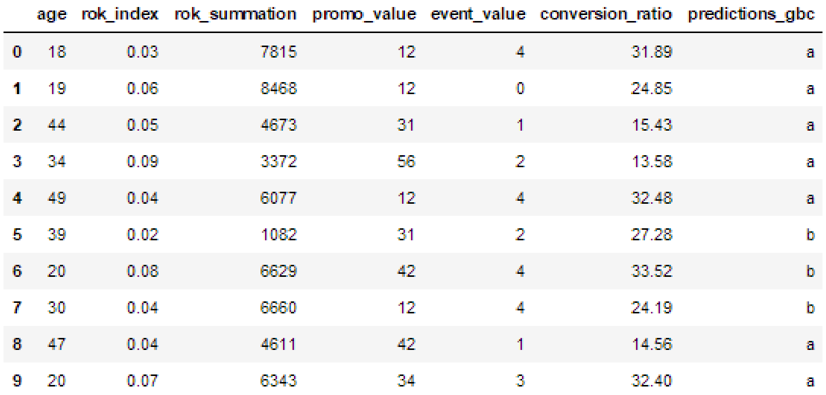
	age	rok_index	rok_summation	promo_value	event_value	conversion_ratio	predictions_dt
0	18	0.03	7815	12	4	31.89	b
1	19	0.06	8468	12	0	24.85	b
2	44	0.05	4673	31	1	15.43	a
3	34	0.09	3372	56	2	13.58	b
4	49	0.04	6077	12	4	32.48	b
5	39	0.02	1082	31	2	27.28	a
6	20	0.08	6629	42	4	33.52	b
7	30	0.04	6660	12	4	24.19	b
8	47	0.04	4611	42	1	14.56	b
9	20	0.07	6343	34	3	32.40	b

d. Prediction on test data using Random Forest:



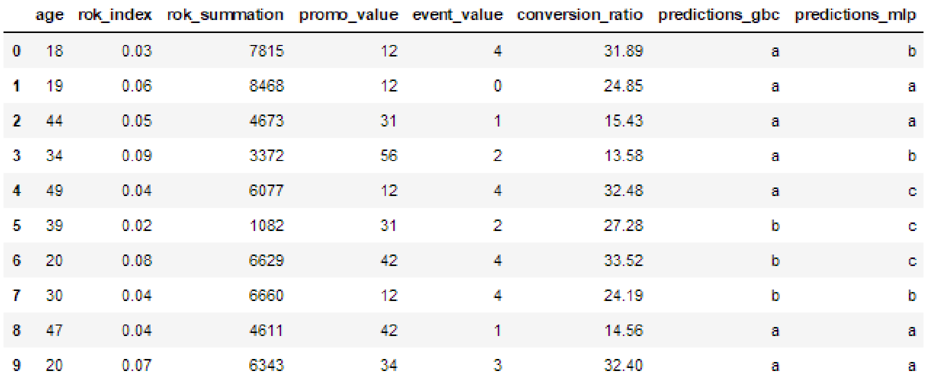
	age	rok_index	rok_summation	promo_value	event_value	conversion_ratio	predictions_rfc
0	18	0.03	7815	12	4	31.89	b
1	19	0.06	8468	12	0	24.85	a
2	44	0.05	4673	31	1	15.43	a
3	34	0.09	3372	56	2	13.58	a
4	49	0.04	6077	12	4	32.48	b
5	39	0.02	1082	31	2	27.28	a
6	20	0.08	6629	42	4	33.52	b
7	30	0.04	6660	12	4	24.19	b
8	47	0.04	4611	42	1	14.56	a
9	20	0.07	6343	34	3	32.40	b

e. Prediction on test data using Gradient boosting Classifier:



	age	rok_index	rok_summation	promo_value	event_value	conversion_ratio	predictions_gbc
0	18	0.03	7815	12	4	31.89	a
1	19	0.06	8468	12	0	24.85	a
2	44	0.05	4673	31	1	15.43	a
3	34	0.09	3372	56	2	13.58	a
4	49	0.04	6077	12	4	32.48	a
5	39	0.02	1082	31	2	27.28	b
6	20	0.08	6629	42	4	33.52	b
7	30	0.04	6660	12	4	24.19	b
8	47	0.04	4611	42	1	14.56	a
9	20	0.07	6343	34	3	32.40	a

f. Prediction on test data using MLP:

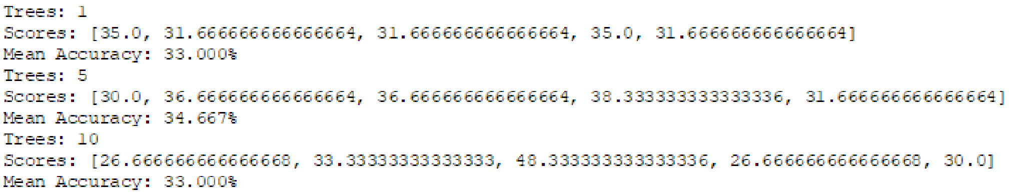


	age	rok_index	rok_summation	promo_value	event_value	conversion_ratio	predictions_gbc	predictions_mlp
0	18	0.03	7815	12	4	31.89	a	b
1	19	0.06	8468	12	0	24.85	a	a
2	44	0.05	4673	31	1	15.43	a	a
3	34	0.09	3372	56	2	13.58	a	b
4	49	0.04	6077	12	4	32.48	a	c
5	39	0.02	1082	31	2	27.28	b	c
6	20	0.08	6629	42	4	33.52	b	c
7	30	0.04	6660	12	4	24.19	b	b
8	47	0.04	4611	42	1	14.56	a	a
9	20	0.07	6343	34	3	32.40	a	a

2. Model from scratch without using any Library, ML Packages and Frameworks:

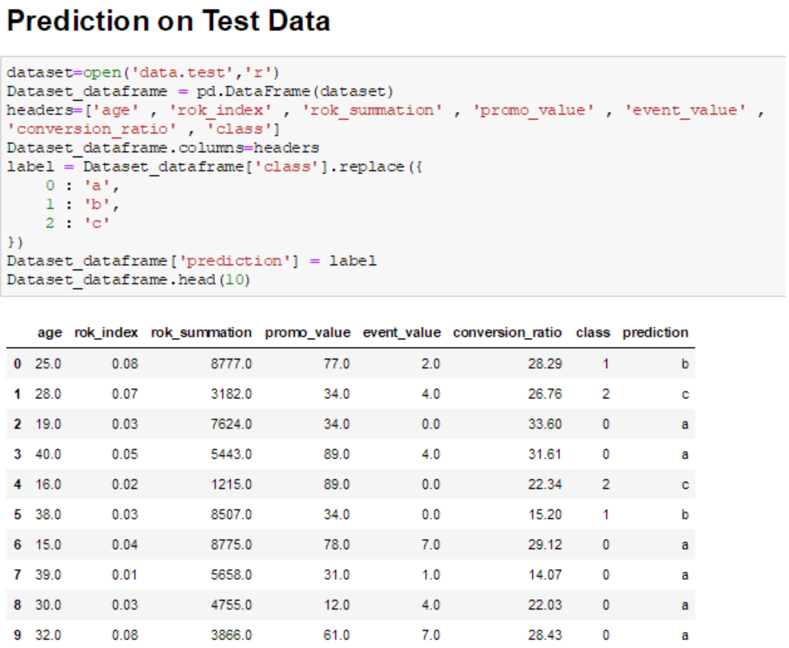
a. Random Forest Algorithm from scratch:

Accuracy:



```
Trees: 1
Scores: [35.0, 31.666666666666664, 31.666666666666664, 35.0, 31.666666666666664]
Mean Accuracy: 33.000%
Trees: 5
Scores: [30.0, 36.666666666666664, 36.666666666666664, 38.333333333333336, 31.666666666666664]
Mean Accuracy: 34.667%
Trees: 10
Scores: [26.666666666666668, 33.333333333333333, 48.333333333333336, 26.666666666666668, 30.0]
Mean Accuracy: 33.000%
```

Prediction on test data:

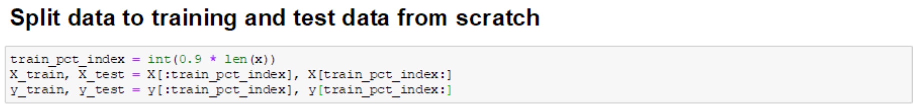


Prediction on Test Data

```
dataset=open('data.test','r')
Dataset_dataframe = pd.DataFrame(dataset)
headers=['age' , 'rok_index' , 'rok_summation' , 'promo_value' , 'event_value' ,
'conversion_ratio' , 'class']
Dataset_dataframe.columns=headers
label = Dataset_dataframe['class'].replace({
    0 : 'a',
    1 : 'b',
    2 : 'c'
})
Dataset_dataframe['prediction'] = label
Dataset_dataframe.head(10)
```

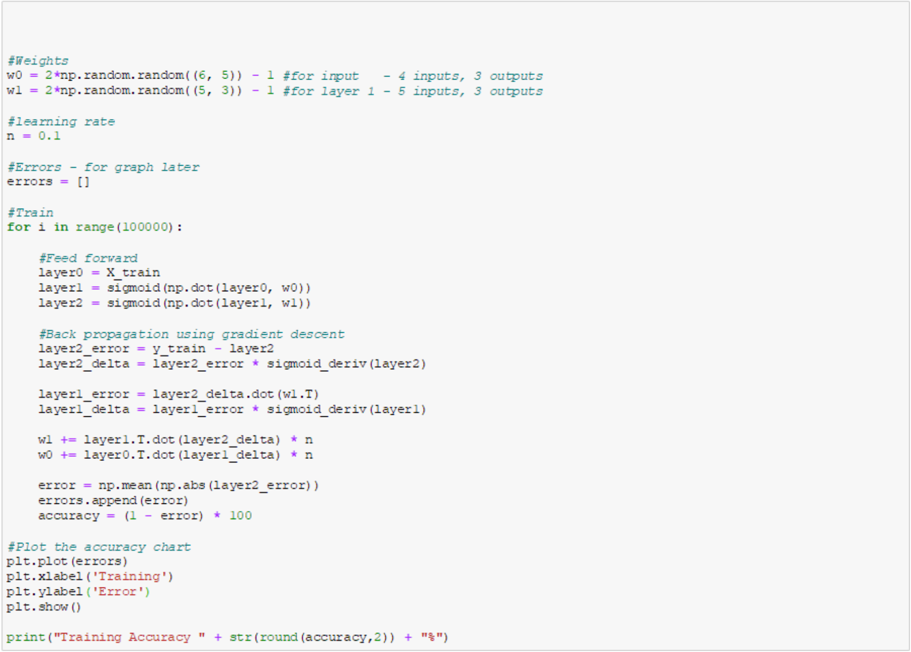
	age	rok_index	rok_summation	promo_value	event_value	conversion_ratio	class	prediction
0	25.0	0.08	8777.0	77.0	2.0	28.29	1	b
1	28.0	0.07	3182.0	34.0	4.0	26.76	2	c
2	19.0	0.03	7624.0	34.0	0.0	33.60	0	a
3	40.0	0.05	5443.0	89.0	4.0	31.61	0	a
4	16.0	0.02	1215.0	89.0	0.0	22.34	2	c
5	38.0	0.03	8507.0	34.0	0.0	15.20	1	b
6	15.0	0.04	8775.0	78.0	7.0	29.12	0	a
7	39.0	0.01	5658.0	31.0	1.0	14.07	0	a
8	30.0	0.03	4755.0	12.0	4.0	22.03	0	a
9	32.0	0.08	3866.0	61.0	7.0	28.43	0	a

b. Multilayer Perceptron (MLP):



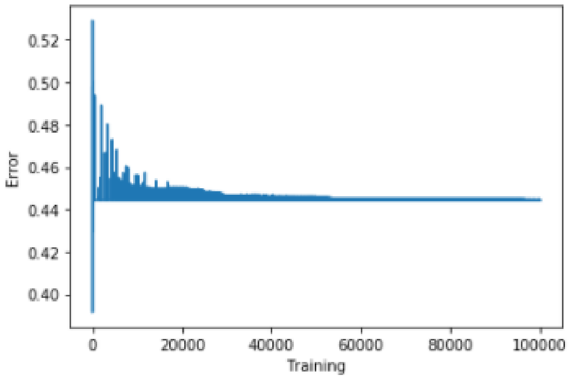
Split data to training and test data from scratch

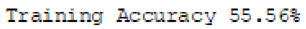
```
train_pct_index = int(0.9 * len(x))
X_train, X_test = X[:train_pct_index], X[train_pct_index:]
y_train, y_test = y[:train_pct_index], y[train_pct_index:]
```



```
#Weights
w0 = 2*np.random.random((6, 5)) - 1 #for input   - 4 inputs, 3 outputs
w1 = 2*np.random.random((5, 3)) - 1 #for layer 1 - 5 inputs, 3 outputs

#learning rate
n = 0.1

#Errors - for graph later
errors = []

#Train
for i in range(100000):

    #Feed forward
    layer0 = X_train
    layer1 = sigmoid(np.dot(layer0, w0))
    layer2 = sigmoid(np.dot(layer1, w1))

    #Back propagation using gradient descent
    layer2_error = y_train - layer2
    layer2_delta = layer2_error * sigmoid_deriv(layer2)

    layer1_error = layer2_delta.dot(w1.T)
    layer1_delta = layer1_error * sigmoid_deriv(layer1)

    w1 += layer1.T.dot(layer2_delta) * n
    w0 += layer0.T.dot(layer1_delta) * n

    error = np.mean(np.abs(layer2_error))
    errors.append(error)
    accuracy = (1 - error) * 100

#Plot the accuracy chart
plt.plot(errors)
plt.xlabel('Training')
plt.ylabel('Error')
plt.show()

print("Training Accuracy " + str(round(accuracy,2)) + "%")
```

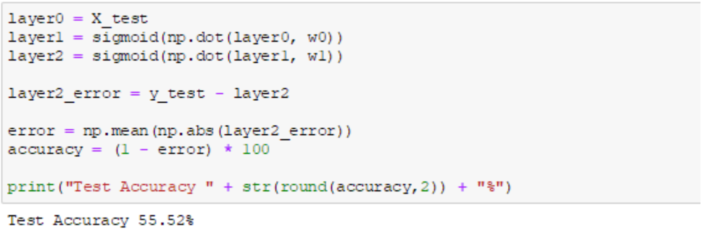
Training Accuracy:





```
Training Accuracy 55.56%
```

Test Accuracy:



```
layer0 = X_test
layer1 = sigmoid(np.dot(layer0, w0))
layer2 = sigmoid(np.dot(layer1, w1))

layer2_error = y_test - layer2

error = np.mean(np.abs(layer2_error))
accuracy = (1 - error) * 100

print("Test Accuracy " + str(round(accuracy,2)) + "%")
```

```
Test Accuracy 55.52%
```

3. Argument on the performances of the Models:

Since the most of data are not showing any normal pattern to classify correctly, therefore we have got very low accuracy for all classification algorithm. When we used built-in library function and classification library and frameworks, for every classifier we have got less than 50% even 40% accuracy. On the other hand, by developing two classification algorithm i.e random forest and MLP from scratch, we have got more than 50% accuracy. So, our build from scratch performs better than using built-in classification algorithm. In conclusion, for less amount of data it would be better build a prediction or classification model from scratch instead of using build-in libraries and classification algorithm.