Retail Manufacturing Analysis using Machine Learning Techniques.

MSc Research Project
Data Analytics

Meet Sangoi
Student ID: X21207526

School of Computing
National College of Ireland

Supervisor: Arjun Chikkankod

National College of Ireland

Project Submission Sheet – 2022/2023

Student Name:	MEET DEEPEN SANGOI
Student ID:	X21207526
Programme:	MSc in Data Analytics **Year:** 2023
Module:	M.Sc. Research Project
Lecturer:	
Submission Due Date:	14-08-2023
Project Title:	Retail Manufacturing Analysis using Machine Learning techniques.
Word Count:	424

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
<u>ALL</u> internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	MEET
Date:	13-08-2023

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed to successfully complete the year. **Any project/assignment not submitted will be marked as a failure.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual: Retail Manufacturing Analysis using Machine Learning Techniques.

Meet Sangoi

x21207526

1 Introduction

I have prepared a manual configuration that delivers a survey of the ‘hardware devices’, ‘software’, and ‘programming skills’ mandatory to carry out the “master’s research project” 'Retail Manufacturing Analytics Using Machine Learning'. It also provides details on the required libraries. The final part of this document contains the code and main output of all runs, results, and evaluation steps.

2 Hardware requirements for research work

A “Lenovo laptop with a 64-bit operating system” is used for the environment setup.

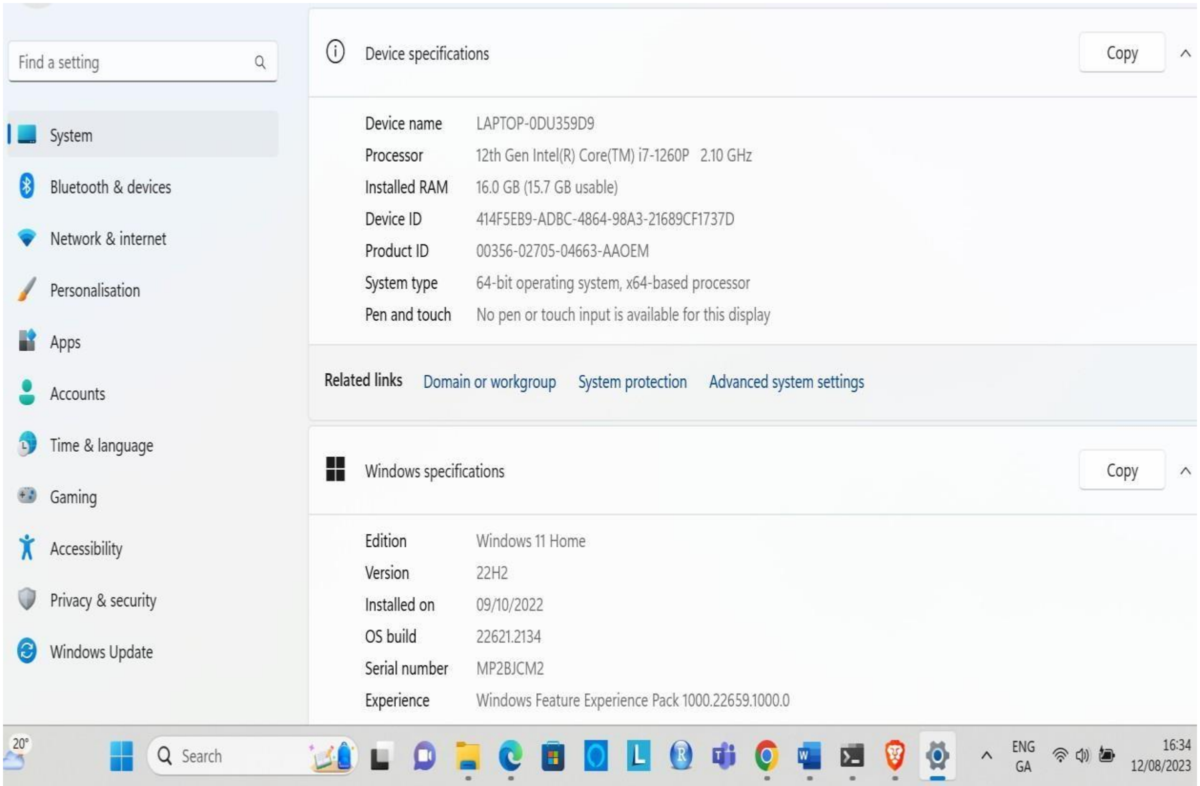

Fig.1. Monitor and Window Description.

The above configuration device "LAPTOP-0DU359D9" is powered by the "12th Gen Intel Core i7-1260P processor" and offers a base clock speed of 2.10 GHz. It has "16.0 GB" of RAM, of which 15.7 GB is available for system operation. I have observed some limitations that need to be checked. Limitations include high execution time in the process train each.

model and the various errors encountered while doing super-tweaking of project settings using super-like libraries.

3 Software required for preparing the analysis.

These scripts were inputted into and executed from a Jupyter book. An integrated development environment (IDE) for writing Python scripts is called Jupyter Book. The data was recorded in a CSV file and retained inside the framework because Jupyter Book may access the dataset directly and run the application within the framework. Open a program in the same registry to pre-install all Python libraries as well as more sophisticated learning systems like TensorFlow, Keras, and sklearn before you can start Jupyter Book.

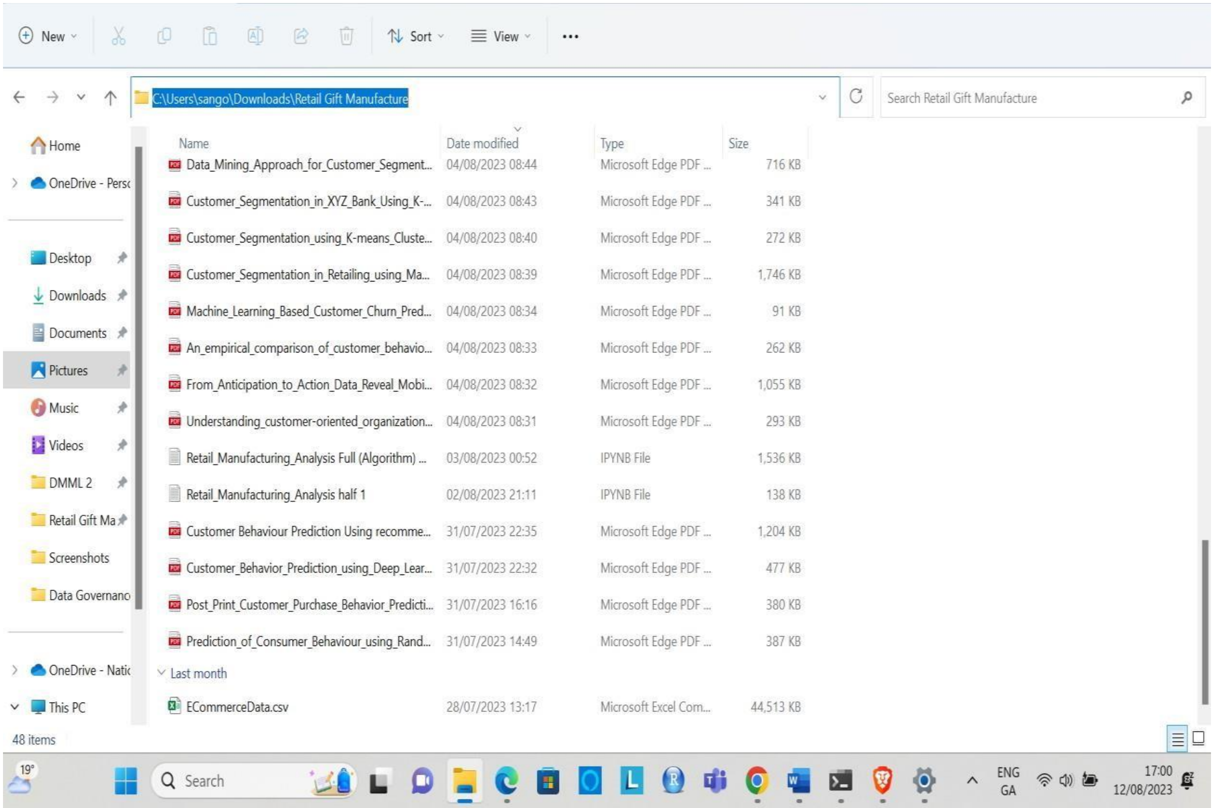

Fig.2. Path in Laptop.

The below fig.3 manifests the important python programming libraries which have been executed in the code.

```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib as mpl
        import matplotlib.pyplot as plt
        import seaborn as sns
        import datetime, nltk, warnings
        import matplotlib.cm as cm
        import itertools
        from pathlib import Path
        from sklearn.preprocessing import StandardScaler
        from sklearn.cluster import KMeans
        from sklearn.metrics import silhouette_samples, silhouette_score
        from sklearn import preprocessing, model_selection, metrics, feature_selection
        from sklearn.model_selection import GridSearchCV, learning_curve
        from sklearn.svm import SVC
        from sklearn.metrics import confusion_matrix
        from sklearn import neighbors, linear_model, svm, tree, ensemble
        from wordcloud import WordCloud, STOPWORDS
        from sklearn.ensemble import AdaBoostClassifier
        from sklearn.decomposition import PCA
        from IPython.display import display, HTML
        import plotly.graph_objs as go
        from plotly.offline import init_notebook_mode,iplot
        init_notebook_mode(connected=True)
        warnings.filterwarnings("ignore")
        plt.rcParams["patch.force_edgecolor"] = True
        plt.style.use('fivethirtyeight')
        mpl.rc('patch', edgecolor = 'dimgray', linewidth=1)
        %matplotlib inline
```

Fig.3. Installed libraries.

Now here, I will obtain the data which is visible in fig.4.

```
In [3]: #Step 1:- Data Preparation
        df_products = pd.read_csv('ECommerceData.csv',encoding="ISO-8859-1")

In [4]: df_products.head() # Display first 5 rows

Out[4]:
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

```
In [5]: print('Dataframe dimensions:', df_products.shape) #Determines the rows and columns of the dataframe

        Dataframe dimensions: (541909, 8)
```

Fig.4. Obtaining the data.

Data Preprocessing

Checking Null values.

In [7]:
```
#Checking for null values
columns_info = pd.DataFrame(df_products.dtypes).T.rename(index={0:'Columns datatype:-'})
columns_info = columns_info.append(pd.DataFrame(df_products.isnull().sum()).T.rename(index={0:'Null values (Count):-'}))
columns_info = columns_info.append(pd.DataFrame(df_products.isnull().sum()/df_products.shape[0]*100).T.
                                   rename(index={0:'Null values (Percentage):-'}))
display(columns_info)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
Columns datatype:-	object	object	object	int64	datetime64[ns]	float64	float64	object
Null values (Count):-	0	0	1454	0	0	0	135080	0
Null values (Percentage):-	0.0	0.0	0.268311	0.0	0.0	0.0	24.926694	0.0

In [8]:
```
#Removed the rows where CustomerID has Nulls
df_products.dropna(axis = 0, subset = ['CustomerID'], inplace = True)
print('Dataframe dimensions:-', df_products.shape)
```

```
Dataframe dimensions:- (406829, 8)
```

Dropping Duplicate values.

In [10]:
```
print(f'Checking for duplicate records:- {df_products.duplicated().sum()}')
```

```
Checking for duplicate records:- 5225
```

In [11]:
```
#Dropping duplicate values
df_products.drop_duplicates(inplace = True)
```

In [12]:
```
print('Dataframe dimensions:-', df_products.shape)
```

```
Dataframe dimensions:- (401604, 8)
```

Orders Per country

```
In [14]: #From below map we came to that most customers are from UK
         product_data = dict(type='choropleth',
         locations = countries.index,
         locationmode = 'country names', z = countries,
         text = countries.index, colorbar = {'title':'Order number'},
         colorscale=[[0, 'rgb(224,255,255)'],
                     [0.01, 'rgb(166,206,227)'], [0.02, 'rgb(31,120,180)'],
                     [0.03, 'rgb(178,223,138)'], [0.05, 'rgb(51,160,44)'],
                     [0.10, 'rgb(251,154,153)'], [0.20, 'rgb(255,255,0)'],
                     [1, 'rgb(227,26,28)']],
         reversescale = False)

         layout = dict(title='Number of orders per country',
         geo = dict(showframe = True, projection={'type':'mercator'}))

         choromap = go.Figure(data = [product_data], layout = layout)
         iplot(choromap, validate=False)
```

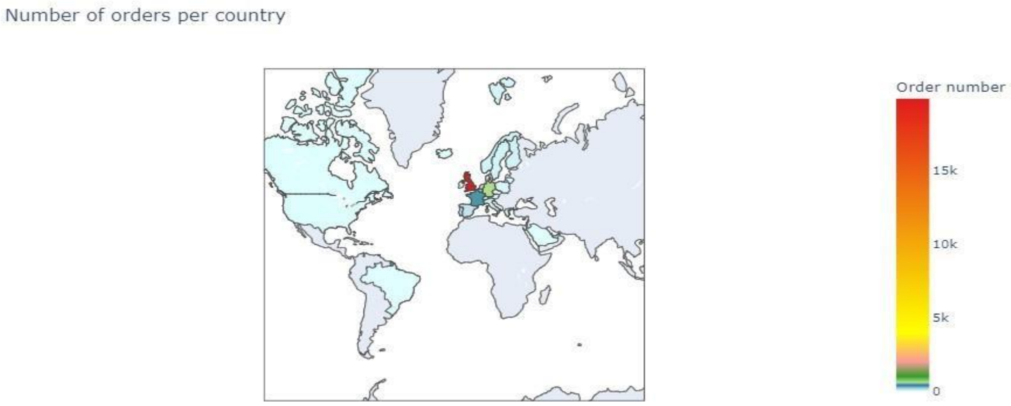

Fig.5. Country wise individual order.

Total number of orders that got cancelled.

```
In [18]: products_per_transaction['Orders_Canceled'] = products_per_transaction['InvoiceNo'].apply(lambda x:int('C' in x))
         display(products_per_transaction.head())
```

	CustomerID	InvoiceNo	Number of products	Orders_Canceled
0	12346.0	541431	1	0
1	12346.0	C541433	1	1
2	12347.0	537626	31	0
3	12347.0	542237	29	0
4	12347.0	549222	24	0

```
In [19]: products_per_transaction_total['Orders_Canceled'] = products_per_transaction_total['InvoiceNo'].apply(lambda x:int('C' in x))
         display(products_per_transaction_total.head())
```

	CustomerID	InvoiceNo	Number of products	Orders_Canceled
6810	14096.0	576339	542	0
6812	14096.0	579196	533	0
6813	14096.0	580727	529	0
6811	14096.0	578270	442	0
6808	14096.0	573576	435	0

```
In [20]: #Total orders that got canceled

         n1 = products_per_transaction_total['Orders_Canceled'].sum()
         n2 = products_per_transaction_total.shape[0]
         print(f'Number of orders canceled: {n1}/{n2} ({n1/n2*100}%)')

         #Number of cancellations is quite large (16% of the total number of transactions)

Number of orders canceled: 3654/22190 (16.466876971608833%)
```

Fig.6. Total number of cancelled orders.

4 Model Buildings

1. Support Vector Machine Model:

```
In [94]: X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, train_size = 0.8)

In [95]: #Support Vector Machine
         svc = Class_Fit(clf = svm.LinearSVC)
         svc.grid_search(parameters = [{'C':np.logspace(-2,2,10)}], Kfold = 5)

In [96]: svc.grid_fit(X = X_train, Y = Y_train)

In [97]: svc.grid_predict(X_test, Y_test)

         Precision: 76.45 %
```

Fig.7. SVM Model

```
In [99]: class_names = [i for i in range(11)]
         cnf_matrix = confusion_matrix(Y_test, svc.predictions)
         np.set_printoptions(precision=2)
         plt.figure(figsize = (8,8))
         plot_confusion_matrix(cnf_matrix, classes=class_names, normalize = False, title='Confusion matrix')

         Confusion matrix, without normalization
```

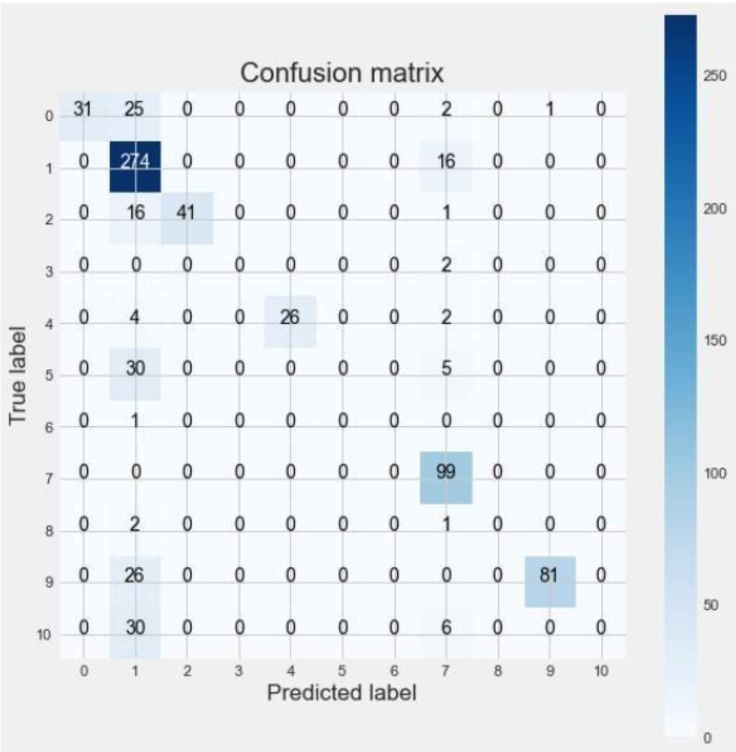

Fig.8. Confusion matrix for SVM

2. K-Nearest Neighbors:

```
In [104]: #k-Nearest Neighbors

knn = Class_Fit(clf = neighbors.KNeighborsClassifier)
knn.grid_search(parameters = [{'n_neighbors': np.arange(1,50,1)}], Kfold = 5)
knn.grid_fit(X = X_train, Y = Y_train)
knn.grid_predict(X_test, Y_test)

Precision: 81.72 %
```

Fig.9. KNN model

3. Decision tree:

```
In [106]: #Decision Tree
tr = Class_Fit(clf = tree.DecisionTreeClassifier)
tr.grid_search(parameters = [{'criterion' : ['entropy', 'gini'], 'max_features' :['sqrt', 'log2']}], Kfold = 5)
tr.grid_fit(X = X_train, Y = Y_train)
tr.grid_predict(X_test, Y_test)

Precision: 83.66 %
```

Fig.10. Decision tree model

4. Random Forest:

```
In [108]: #Random Forest
rf = Class_Fit(clf = ensemble.RandomForestClassifier)
param_grid = {'criterion' : ['entropy', 'gini'], 'n_estimators' : [20, 40, 60, 80, 100],
               'max_features' :['sqrt', 'log2']}
rf.grid_search(parameters = param_grid, Kfold = 5)
rf.grid_fit(X = X_train, Y = Y_train)
rf.grid_predict(X_test, Y_test)

Precision: 90.30 %
```

Fig.11. Random forest model

5. Logistic regression:

```
In [102]: #Logistic Regression
lr = Class_Fit(clf = linear_model.LogisticRegression)
lr.grid_search(parameters = [{'C':np.logspace(-2,2,20)}], Kfold = 5)
lr.grid_fit(X = X_train, Y = Y_train)
lr.grid_predict(X_test, Y_test)

Precision: 89.61 %
```

Fig.12. Logistic regression model

6. Gradient Boosting:

```
In [112]: #Gradient Boosting Classifier
gb = Class_Fit(clf = ensemble.GradientBoostingClassifier)
param_grid = {'n_estimators' : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]}
gb.grid_search(parameters = param_grid, Kfold = 5)
gb.grid_fit(X = X_train, Y = Y_train)
gb.grid_predict(X_test, Y_test)

Precision: 89.75 %
```

Fig.13. Gradient boosting model

Voting classifier

In [114]:
```
rf_best  = ensemble.RandomForestClassifier(**rf.grid.best_params_)
gb_best  = ensemble.GradientBoostingClassifier(**gb.grid.best_params_)
svc_best = svm.LinearSVC(**svc.grid.best_params_)
tr_best  = tree.DecisionTreeClassifier(**tr.grid.best_params_)
knn_best = neighbors.KNeighborsClassifier(**knn.grid.best_params_)
lr_best  = linear_model.LogisticRegression(**lr.grid.best_params_)
```

In [115]:
```
votingC = ensemble.VotingClassifier(estimators=[('rf', rf_best),('gb', gb_best),
                                                ('knn', knn_best)], voting='soft')
```

In [116]:
```
votingC = votingC.fit(X_train, Y_train)
```

In [117]:
```
predictions = votingC.predict(X_test)
print("Precision: {:.2f} % ".format(100*metrics.accuracy_score(Y_test, predictions)))
```

```
Precision: 90.03 %
```

Fig.14. Voting classifier models

Testing Prediction

In [124]:
```
classifiers = [(svc, 'Support Vector Machine'),
                (lr, 'Logistic Regression'),
                (knn, 'k-Nearest Neighbors'),
                (tr, 'Decision Tree'),
                (rf, 'Random Forest'),
                (gb, 'Gradient Boosting')]

for clf, label in classifiers:
    print(25*'_', '\n{}'.format(label))
    result = clf.grid_predict_precision(X, Y)
    df_result.loc[len(df_result.index)] = [label, result]
```

```
_________________________
Support Vector Machine
Precision: 62.68 %
_________________________
Logistic Regression
Precision: 75.11 %
_________________________
k-Nearest Neighbors
Precision: 67.19 %
_________________________
Decision Tree
Precision: 71.23 %
_________________________
Random Forest
Precision: 74.87 %
_________________________
Gradient Boosting
Precision: 74.48 %
```

Fig.15. Final Testing Prediction

```
In [127]: import matplotlib.pyplot as plt

          def addlabels(x,y):
              for i in range(len(x)):
                  plt.text(i,y[i],y[i])

          plt.figure(figsize=(20, 7))
          plt.bar(df_result.Algorithms, df_result.Precision, color='green', width=0.4, align='center')
          addlabels(df_result.Algorithms, df_result.Precision)
          plt.title('Precision for different alogirthms')
          plt.xlabel('Algorithms')
          plt.ylabel('Precision')
          plt.show()
```

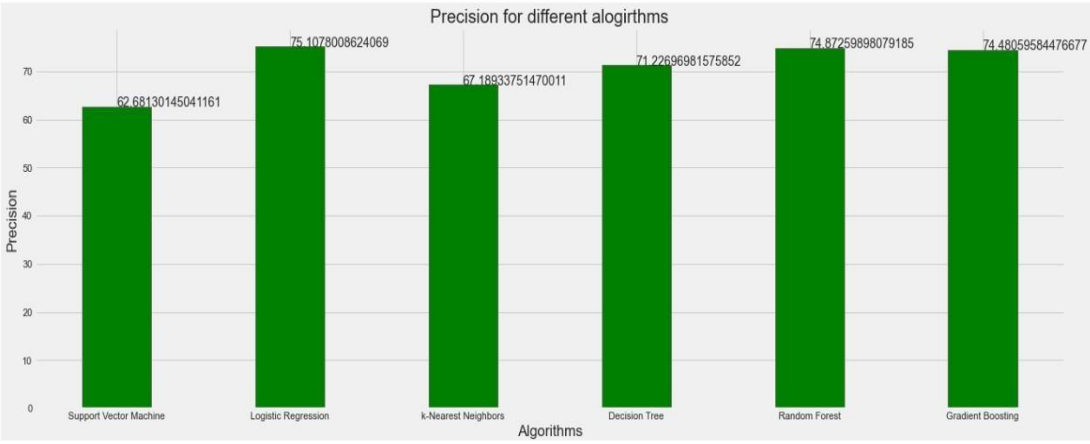

Fig.16. Bar plot for all model’s accuracy.

Therefore, by seeing all models here and by visualizing the accuracies and graph, I will forecast the customer’s needs through Logistic regression model.