ADVANCED ACADEMIC CENTER

A CENTER FOR INTER-DISCIPLINARY RESEARCH
2020-21
TITLE

# “PRODUCT DEMAND PREDICTION”

SUPERVISED BY

PADMINI KOUSALYA MADHIRA

GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
AUTONOMOUS

# Advanced Academic Center

**( A Center For Inter-Disciplinary Research )**

This is to certify that the project titled

**“PRODUCT DEMAND PREDICTION”**

is a bonafide work carried out by the following students in partial fulfilment of the requirements for Advanced Academic Center intern, submitted to the chair, AAC during the academic year 2020-21.

| NAME | ROLL N0 | BRANCH |
|---|---|---|
| CHINNOLLA KOTESHWAR | 21241A0476 | ECE |
| KANISHK BANSAL | 21241A1226 | IT |
| KARTHIKEYA KOLLURI | 21241A1294 | IT |
| THANOJ AILA | 21241A1266 | IT |
| KRUTHIK REDDY JUVVENTHULA | 21241A66F6 | CSM |

This work was not submitted or published earlier for any study

Dr/Ms./
Mr.

Project
Supervisor

Dr.B.R.K.Reddy
Program Coordinator

Dr.Ramamurthy Suri
Associate Dean,AAC


# ACKNOWLEDGEMENTS

We express our deep sense of gratitude to our respected Director, Gokaraju Rangaraju Institute of Engineering and Technology, for the valuable guidance and for permitting us to carry out this project.

With immense pleasure, we extend our appreciation to our respected Principal, for permitting us to carry out this project.

We are thankful to the Associate Dean, Advanced Academic Centre, for providing us an appropriate environment required for the project completion.

We are grateful to our project supervisor who spared valuable time to influence us with their novel insights.

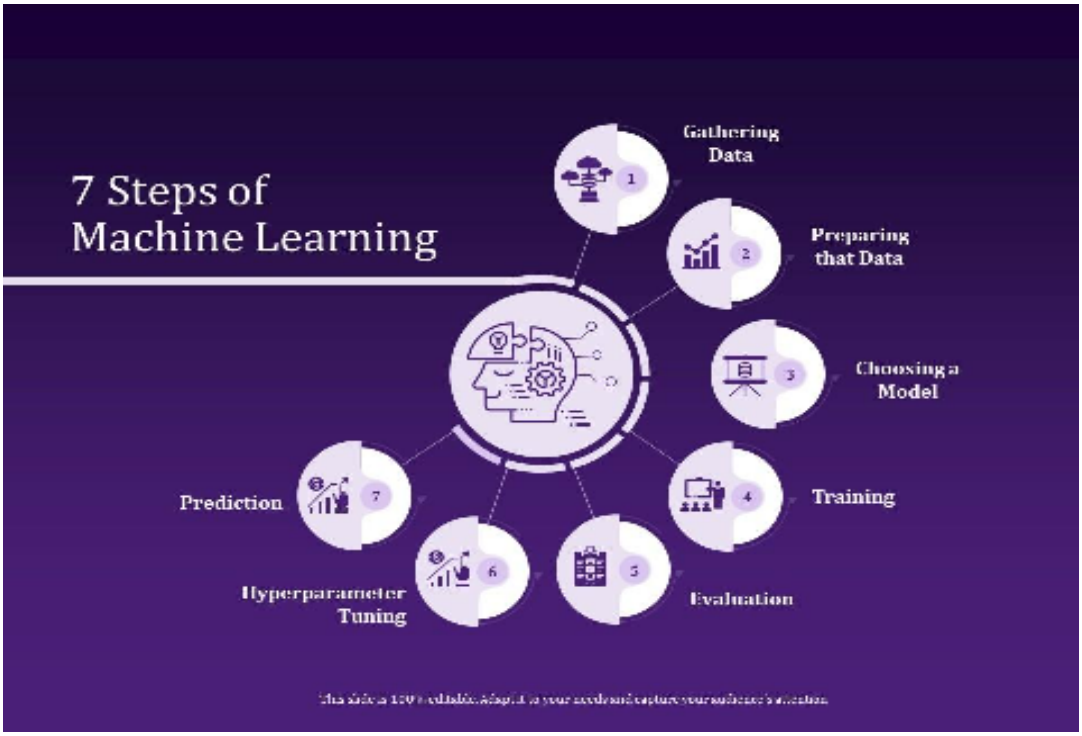
We are indebted to all the above mentioned people without whom we would not have concluded the project.

## ABSTRACT:

The key to success in today's business is controlling the retail supply chain. Predicting customer demand is very essential for supply chain management. The perfect prediction has an effective impact on earning a profit., storage., lost profit., sales amount and consumer attraction. This article will produce a new method-using machine learning that will help for accurate prediction. This method collects the previous data of a store and analyses that data. Gather the important information, process that data and get prepared for using it. Applying related algorithms towards the process data. We know K-Nearest Neighbor, Support Vector Machine, Gaussian Naive Bayes, Random Forest, Decision Tree Classifier and regressions have recently used an algorithm for prediction. We collect real-life data from the market. This paper was made with the combination of shop position, month and occasion on that month and other related data. Our country's geographical area has an impact on prediction, which we discuss in our research. Our model produces a tentative demand for a particular product. This estimation helps retailers and their businesses. After making a data set and applying appropriate algorithms, we will find different results and accuracy of different used algorithms. Compare them with others, we find out Gaussian Naive Bayes has the best accuracy. This helps to estimate the accurate product demand for a shop.

## Introduction:

- Market Analysis is important and crucial for any money making business. In order to sustain or grow the business should know their customers and demand so that they can amend their plans accordingly and be profitable. With this PDF model we're trying to make an analysis of demand for certain products and give an idea about their product demand.
- In order to make any sort of analysis, we need to figure out how much demand there is for certain products and what the prices of those products are we tried to do this exact same thing with our project.
- We tried to predict the demand percentage and co-relate with the demand and supply chain.



## Product demand prediction using machine learning:

- Product demand prediction using machine learning by random forest for the most accurate result.
- Random forest (RF) is a type of meta-learner that uses a number of decision trees for both classification and regression problems (Breiman, 2001). The features and samples are drawn randomly for every tree in the forest and these trees are trained independently. Each tree is constructed with a bootstrap sampling method. Bootstrapping relies on sampling with replacement. Given a dataset D with N samples, a training data set of size N is created by sampling from D with replacement. The remaining samples in D that are not in the training set are separated as the test set. This kind of sampling is called bootstrap sampling.
- Samples in the test set are called out-of-bag data. On the other hand, every tree has different features which are selected randomly. While selecting nodes in the tree, only a subset of the features are selected and the best one is chosen as a separator node from this subset. Then this process continues recursively until a certain error rate is reached. Each tree is grown independently to reach the specified error rate. For instance, stock feature is chosen as the best separator node among the other randomly selected features, and likewise price feature is chosen as second best node for the first tree in Figure 3. This tree is constructed with two nodes such as stock and price, whereas TREE N has four nodes and some of them are different from TREE 1.
- Due to the bootstrapping sampling method, there is no need to use cross-validation or separate datasets for training and testing. This process is done internally. In this project, minimum root mean squared error was achieved by using random forest with 20 trees in the first level.
-

## Project Workflow:

- Defined objectives
- Collected data from various sources
- Changes made to raw data
- Imported the dataset
- Reading Data using Pandas
- Encoding Data
- Divided into training and test data
- Using different Regression models
- Plotting Graphs for the outputs.
- We got the accuracy about 93.5% using Random forest model we can use other models for higher accuracy depending on the data set taken.

CODE:

1. IMPORTING LIBRARIES

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GridSearchCV
```

## 2. READING DATA USING PANDAS

```
data = pd.read_csv('prod_.csv')
data.head(15)
```

| | product | year | expiri duretion in years | month | no. of new products bought | no of products left for selling | no. of products sold | prodects left | no of products to be bought | demand % |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | colgate | 2019 | 2.0 | 1 | 60 | 60 | 46 | 0 | 46.0 | 76.666667 |
| 1 | colgate | 2019 | 2.0 | 2 | 32 | 46 | 38 | 14 | 24.0 | 118.750000 |
| 2 | colgate | 2019 | 2.0 | 3 | 40 | 62 | 47 | 22 | 25.0 | 117.500000 |
| 3 | colgate | 2019 | 2.0 | 4 | 36 | 51 | 40 | 15 | 25.0 | 111.111111 |
| 4 | colgate | 2019 | 2.0 | 5 | 56 | 67 | 58 | 11 | 47.0 | 103.571429 |
| 5 | colgate | 2019 | 2.0 | 6 | 41 | 50 | 39 | 9 | 30.0 | 95.121951 |
| 6 | colgate | 2019 | 2.0 | 7 | 58 | 69 | 51 | 11 | 40.0 | 87.931034 |
| 7 | colgate | 2019 | 2.0 | 8 | 31 | 47 | 47 | 16 | 31.0 | 151.612903 |
| 8 | colgate | 2019 | 2.0 | 9 | 100 | 100 | 56 | 0 | 56.0 | 56.000000 |
| 9 | colgate | 2019 | 2.0 | 10 | 0 | 44 | 44 | 44 | 44.0 | 44.000000 |
| 10 | colgate | 2019 | 2.0 | 11 | 38 | 38 | 16 | 0 | 16.0 | 42.105263 |
| 11 | colgate | 2019 | 2.0 | 12 | 30 | 52 | 48 | 22 | 26.0 | 160.000000 |
| 12 | colgate | 2020 | 2.0 | 1 | 72 | 76 | 56 | 4 | 52.0 | 77.777778 |
| 13 | colgate | 2020 | 2.0 | 2 | 78 | 98 | 86 | 20 | 66.0 | 110.256410 |
| 14 | colgate | 2020 | 2.0 | 3 | 60 | 72 | 59 | 12 | 47.0 | 98.333333 |

```
data['product'].unique()
```

```
array(['colgate', 'gemini  tea powder', 'suger', 'freedom sunflower oil',
       'life boy soap', 'santoor'], dtype=object)
```

## 3. ENCODING DATA

```
ohe = OneHotEncoder()
ohe.fit(data[['product']])
ct=make_column_transformer((OneHotEncoder(),[0]),remainder='passthrough')
```

# 4. DIVINDING DATA INTO TRAINING AND TESTING SET

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.simplefilter(action='ignore', category=UserWarning)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 0)
```

```
x_train
```

```
array([['life boy soap', 2019, 8],
       ['gemini  tea powder', 2019, 5],
       ['colgate', 2020, 2],
       ['freedom sunflower oil', 2019, 5],
       ['suger', 2020, 1],
       ['colgate', 2021, 1],
       ['colgate', 2021, 7],
       ['gemini  tea powder', 2021, 1],
       ['gemini  tea powder', 2020, 9],
       ['life boy soap', 2019, 3],
       ['suger', 2021, 1],
       ['colgate', 2020, 8],
       ['life boy soap', 2019, 10],
       ['life boy soap', 2020, 6],
       ['gemini  tea powder', 2020, 7],
```

# 5. REGRESSION MODELS

- LINEAR REGRESSION

```
lr = LinearRegression()
```

MAKING PIPELINE

```
from pandas.core.algorithms import mode
from sklearn import pipeline
pipe=make_pipeline(ct,lr)
pipe.fit(x_train,y_train)
```

```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',
                                                  OneHotEncoder(), [0])])),
                ('linearregression', LinearRegression())])
```

SCORE

```
y_pred= pipe.predict(x_test)
score = r2_score(y_pred,y_test)
score
```

```
0.9052419107148001
```

```
scores=[]
for i in range(1000):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=i)
    dt = LinearRegression()
    piped=make_pipeline(ct,dt)
    piped.fit(x_train,y_train)
    y_pred=piped.predict(x_test)
    scores.append(r2_score(y_test,y_pred))
scores[np.argmax(scores)]
```

```
pipe.predict([['santoor',2021,3]])
```

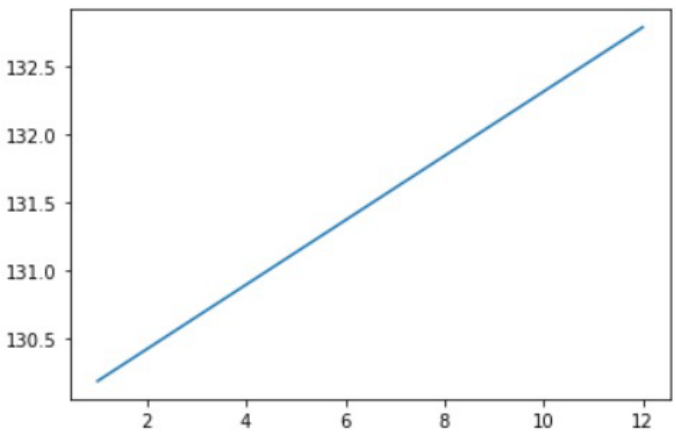
```
array([[153.92103069]])
```

```
c=[]
for i in range(0,12):
  c.append(float(pipe.predict([['colgate',2023,i]])))
```

```
c
```

```
[130.1850150872051,
 130.42168492095516,
 130.65835475469794,
 130.895024588448,
 131.13169442219078,
 131.36836425594083,
 131.6050340896909,
 131.84170392343367,
 132.07837375718373,
 132.31504359093378,
 132.55171342467656,
 132.78838325842662]
```

```
month=data.iloc[:12,[3]].values
plt.plot(month,c)
plt.show()
```




```
warnings.simplefilter(action='ignore', category=UserWarning)
```

# Decision Trees

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import fbeta_score, make_scorer
dt = DecisionTreeRegressor()
parameters = {'max_depth':(1,2,3,4,5,6,7,8,9,10)}
scoring_function = make_scorer(fbeta_score, beta=2)
dtg = GridSearchCV(dt, param_grid=parameters, scoring=scoring_function)
piped = make_pipeline(ct,sc,dtg)
piped.fit(x_train,y_train)
y_pred=piped.predict(x_test)
r2_score(y_test,y_pred)
```

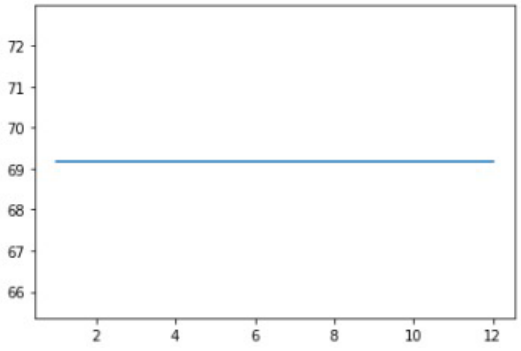
```
0.8086454646920715
```

```
c_=[]
for i in range(0,12):
    c_.append(float(piped.predict([['colgate',2022,i]])))
```

```
c_
```

```
[69.16666666666667,
 69.16666666666667,
 69.16666666666667,
 69.16666666666667,
 69.16666666666667,
 69.16666666666667,
 69.16666666666667,
 69.16666666666667,
 69.16666666666667,
 69.16666666666667,
 69.16666666666667,
 69.16666666666667]
```

```
month=data.iloc[:12,[3]].values
plt.plot(month,c_)
plt.show()
```



# Random Forest

```
from sklearn.ensemble import RandomForestRegressor
estimator = RandomForestRegressor()

piper = make_pipeline(ct,estimator)
piper.fit(x_train, y_train)
y_pred= piper.predict(x_test)
r2_score(y_test, y_pred)
```

```
0.9026506512930348
```

```
piper.predict([['suger',2023,1]])
```

```
array([407.15])
```

```
k=data['product'].unique()
colgate1=[]
for j in range(1,13):
    colgate1.append(float(piper.predict([['colgate',2022,j]])))
```

```
k
```

```
array(['colgate', 'gemini  tea powder', 'suger', 'freedom sunflower oil',
       'life boy soap', 'santoor'], dtype=object)
```

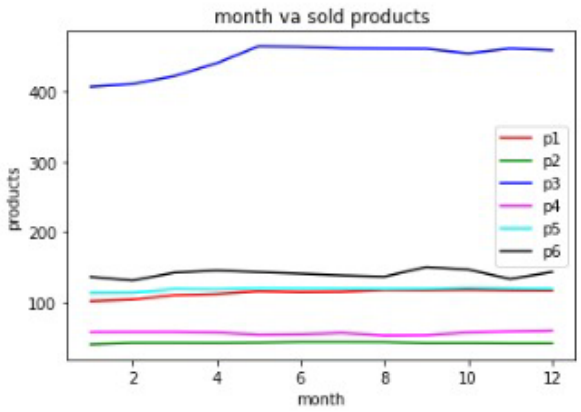
```
p1=[]
p2=[]
p3=[]
p4=[]
p5=[]
p6=[]
for i in range(0,6):
  for j in range(1,13):
    if i==0:
      p1.append(float(piper.predict([[k[i],2022,j]])))
    if i==1:
      p2.append(float(piper.predict([[k[i],2022,j]])))
    if i==2:
      p3.append(float(piper.predict([[k[i],2022,j]])))
    if i==3:
      p4.append(float(piper.predict([[k[i],2022,j]])))
    if i==4:
      p5.append(float(piper.predict([[k[i],2022,j]])))
    if i==5:
      p6.append(float(piper.predict([[k[i],2022,j]])))
```

```
p2
```

```
[39.37,
 41.47,
 41.47,
 41.29,
 41.84,
 42.73,
 42.72,
 42.56,
 41.0,
 41.12,
 40.7,
 40.7]
```

6.GRAPHS

```
month=data.iloc[:12,[3]].values
plt.plot(month,p1,c='red',label='p1')
plt.plot(month,p2,c='green',label='p2')
plt.plot(month,p3,c='blue',label='p3')
plt.plot(month,p4,c='magenta',label='p4')
plt.plot(month,p5,c='cyan',label='p5')
plt.plot(month,p6,c='black',label='p6')
plt.title('month va sold products')
plt.xlabel('month')
plt.ylabel('products')
plt.legend()
plt.show()
```

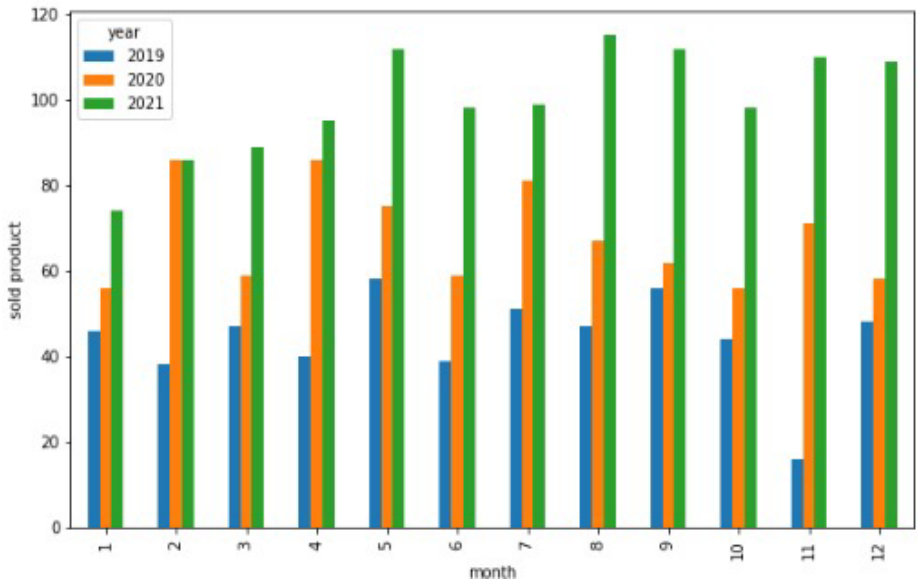



MONTH VS DEMAND

## 7. BAR GRAPHS

```
for i in range(0,6):
  k[i]=k[i].pivot(index='month', columns = 'year',values = 'no. of products sold')
```

```
for i in range(0,6):
  k[i].plot(kind='bar',figsize=(10,6))
  plt.xlabel('month')
  plt.ylabel('sold product')
```




BAR GRAPHS REPRESENTING THE OUTPUT

# FUTURE DEVOLOPMENTS

- To improve the diversity of the datasets taken.
- Improve the web interface to add in data monthly data by the user so that we can keep on improving the data and have accurate predictions.
- Tag the expiry date with the barcode of the product and try

# References


- https://scikit-learn.org/stable/modules/classes.html
- https://www.w3schools.com/html/html_editors.asp
- https://stackoverflow.com/questions