PUNE INSTITUTE OF COMPUTER TECHNOLOGY,
DHANKAWADI PUNE-43.

A Project Report On

Bigmart Sales Prediction

SUBMITTED BY

NAME: Kunal Satish Kunkulol
ROLL NO: 4436
CLASS: BE4

GUIDED BY
Prof. Pravin Patil

CERTIFICATE

This is to certify that **Mr. Kunal Satish Kunkulol**, Roll No. 4436 a student of B.E. (Computer Engineering Department)
Batch 2018-2019, has satisfactorily completed a project report on “**Bigmart Sales Prediction**” under the guidance of **Prof. Pravin Patil** towards the partial fulfillment of the fourth year Computer Engineering Semester I of Savitribai Phule Pune University.

Prof. Pravin Patil	Dr. R.B.Ingle
Project Guide	Head of Department,
	Computer Engineering

ABSTRACT

Nowadays a lot of online transactions occur. Due to such huge number of transactions a lot of data is generated. Now such data can be utilized for building the predictive model. This model will help us to analyze and understand the trends of the data. Now various different general store shops are there which sales multiple products and they have various branches , Bigmart is one such commodity. Bigmart has collected the data of sales of particular product at their 10 different outlets. Thus here building the predictive model will give them the information about how to distribute the products in different outlets based on the past utilization. This will basically reduce the extra-cost and give them advantage over others. Here we basically are building the predictive model for predicting the sales of the particular product in particular outlet.

CONTENTS

1. PROBLEM STATEMENT

The data scientists at BigMart have collected 2013 sales data for 1559 products across 10 stores in different cities. Also, certain attributes of each product and store have been defined. The aim is to build a predictive model and find out the sales of each product at a particular store.Using this model, BigMart will try to understand the properties of products and stores which play a key role in increasing sales. Predictive Models are one of the most common applications of the Data Analytics. Predictive model can predict the different types of things. Predictive models provide the information related to sales which will lead to optimized production of the products.

2. Requirement Specification

1. **Jupyter Notebook**
   The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. The Notebook has support for over 40 programming languages, including Python, R, Julia, and Scala.

2. **Python Libraries(Sklearn,Numpy,Pandas,Matplotlib)**
   NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object and an assortment of routines for fast operations on arrays, including mathematical, logical etc. *pandas* is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

3)Sales Prediction Model:

The two major components of this model. They are

1) Data Cleaning

1. Deleting Rows -This method commonly used to handle the null values. Here, we either delete a particular row if it has a null value for a particular feature and a particular column if it has more than 70-75% of missing values. This method is advised only when there are enough samples in the data set.

2. Replacing with Mean/Median/Mode - This strategy can be applied on a feature which has numeric data like the age of a person or the ticket fare. We can calculate the mean, median or mode of the feature and replace it with the missing values. This is an approximation which can add variance to the data set. But the loss of the data can be negated by this method which yields better results compared to removal of rows and columns.

3. Assigning an Unique Category - A categorical feature will have a definite number of possibilities, such as gender, for example. Since they have a definite number of classes, we can assign another class for the missing values. Here, the features Cabin and Embarked have missing values which can be replaced with a new category

4. Predicting the Missing Values - Using the features which do not have missing values, we can predict the nulls with the help of a machine learning algorithm. This method may result in better accuracy, unless a missing value is expected to have a very high variance.

5. Using Algorithms Which Support Missing Values - KNN is a machine learning algorithm which works on the principle of distance measure. This algorithm can be used when there are nulls present in the dataset.

2) Feature Engineering

Feature engineering is the process of using **domain knowledge** of the data to create features that make machine learning algorithms work. If feature engineering is done correctly, it increases the predictive power of machine learning algorithms by creating features from raw data that help facilitate the machine learning process. Feature Engineering is an art. Feature engineering is the most important art in machine learning which creates the huge difference between a good model and a bad model. Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.

4)SOURCE CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#get_ipython().run_line_magic('matplotlib', 'inline')
from scipy.stats import kurtosis,skew
from sklearn.linear_model import LinearRegression,Ridge
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.tree import DecisionTreeRegressor,ExtraTreeRegressor
from sklearn.ensemble import 
RandomForestRegressor,BaggingRegressor,GradientBoostingRegressor,Extra
TreesRegressor
import xgboost,lightgbm
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_error,mean_squared_error
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import ElasticNet, Lasso,  BayesianRidge, 
LassoLarsIC
from sklearn.ensemble import RandomForestRegressor,  
GradientBoostingRegressor
from sklearn.kernel_ridge import KernelRidge
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import 
RobustScaler,MinMaxScaler,StandardScaler
from sklearn.model_selection import KFold, cross_val_score, 
train_test_split
from sklearn.metrics import mean_squared_error

d1=pd.read_csv('Train_UWu5bXk.csv')
data=d1
d1.shape
data['Item_Fat_Content'].value_counts()
data['Item_Fat_Content']=data['Item_Fat_Content'].map({'LF':0,'Low 
Fat':0,'Regular':1,'low fat':0,'reg':1})
yy=data.groupby('Item_Identifier')['Item_Weight'].mean()
yy=dict(yy)
data['Item_Weight'].fillna(-1.0,inplace=True)
def ff(xx,d):
    if(d==-1.0):
        return yy[xx]
    else:
        return d
```

```
data['Item_Weight']=data.apply(lambda x :
ff(x['Item_Identifier'],x['Item_Weight']),axis=1)
data.isnull().sum()
data.groupby('Item_Type')['Item_Weight'].median()
data['Item_Weight'].fillna(10.0,inplace=True)
data.groupby('Outlet_Type')['Outlet_Size'].value_counts()
data['Outlet_Size'].fillna('Small',inplace=True)
np.where(data.Item_Visibility<=0)
yy=data[data.Item_Visibility>0.0].groupby('Item_Identifier')
['Item_Visibility'].mean()
yy=dict(yy)

def ff(x,y):
    if(x==0.0):
        return yy[y]
    else:
        return x
data['Item_Visibility']=data.apply(lambda x :
ff(x['Item_Visibility'],x['Item_Identifier']),axis=1)
ya=data.groupby('Item_Identifier')['Item_Visibility'].mean()
ya=dict(ya)

def ff(x,y):
    return (x/ya[y])
data['itemimportance']=data.apply(lambda x :
ff(x['Item_Visibility'],x['Item_Identifier']),axis=1)

data['itemimportance'].describe()
data['years']=2018-data['Outlet_Establishment_Year']
del data['Outlet_Establishment_Year']

lb=LabelEncoder()
lb.fit(data['Item_Type'])
data['Item_Type']=lb.transform(data['Item_Type'])
lb=LabelEncoder()


lb.fit(data['Outlet_Identifier'])
data['Outlet_Identifier']=lb.transform(data['Outlet_Identifier'])

lb=LabelEncoder()
lb.fit(data['Outlet_Size'])
data['Outlet_Size']=lb.transform(data['Outlet_Size'])

lb=LabelEncoder()
lb.fit(data['Outlet_Location_Type'])
data['Outlet_Location_Type']=lb.transform(data['Outlet_Location_Type']
)

lb=LabelEncoder()
lb.fit(data['Outlet_Type'])
data['Outlet_Type']=lb.transform(data['Outlet_Type'])
```

```
del data['Item_Identifier']
test=data[8500:]
data=data[0:8500]
yd=data['Item_Outlet_Sales']
del data['Item_Outlet_Sales']
del test['Item_Outlet_Sales']
data.shape,yd.shape,test.shape

yy=yd
#simple linear regression with cross validation of 5 fold
avg=0.0
skf=KFold(n_splits=5)
skf.get_n_splits(data)
for ti,tj in skf.split(data):
    dx,tx=data.iloc[ti],data.iloc[tj]
    dy,ty=yy[ti],yy[tj]
    lm=make_pipeline(MinMaxScaler(),
linear_model.LinearRegression(n_jobs=-1))
    lm.fit(dx,dy)
    yu=np.sqrt(mean_squared_error(y_true=ty,y_pred=lm.predict(tx)))
    avg=avg+yu
    print(yu)

print("AVG  RMSE::",avg/5)
#simple elasticnet regression with cross validation of 5 fold
avg=0.0
skf=KFold(n_splits=5)
skf.get_n_splits(data)
for ti,tj in skf.split(data):
    dx,tx=data.iloc[ti],data.iloc[tj]
    dy,ty=yy[ti],yy[tj]
    lm=make_pipeline(StandardScaler(),
linear_model.ElasticNet(l1_ratio=0.6,alpha=0.001))
    lm.fit(dx,dy)
 yu=np.sqrt(mean_squared_error(y_true=ty,y_pred=lm.predict(tx)))
    avg=avg+yu
    print(yu)
print("AVG  RMSE::",avg/5)

#simple Decision Tree regression with cross validation of 5 fold
avg=0.0
skf=KFold(n_splits=5)
skf.get_n_splits(data)
for ti,tj in skf.split(data):
    dx,tx=data.iloc[ti],data.iloc[tj]
    dy,ty=yy[ti],yy[tj]
    lm=DecisionTreeRegressor(max_depth=5)
    lm.fit(dx,dy)
   yu=np.sqrt(mean_squared_error(y_true=ty,y_pred=lm.predict(tx)))
    avg=avg+yu
    print(yu)
print("AVG  RMSE::",avg/5)
#simple Random Forest Tree regression with cross validation of 5 fold
avg=0.0
skf=KFold(n_splits=5)
skf.get_n_splits(data)
for ti,tj in skf.split(data):
    dx,tx=data.iloc[ti],data.iloc[tj]
    dy,ty=yy[ti],yy[tj]
    lm=RandomForestRegressor(max_depth=5,n_jobs=-1,n_estimators=100)
    lm.fit(dx,dy)
yu=np.sqrt(mean_squared_error(y_true=ty,y_pred=lm.predict(tx)))
    avg=avg+yu
    print(yu)
print("AVG  RMSE::",avg/5)
```

5.Results

1. Linear regression
   RMSE : 1168.017
2. Elastic Net Regression
   RMSE : 1168.016
3. Decision Tree Regerssor
   RMSE : 1095.2

6.CONCLUSION

With this project we have build the predictive model for predict the sales which will help the bigmart for production of the optimised cost of products in each particular project.