**REPORT**

**Warehouseify – Tech Tycoons**

****

by

**Mahima M, Shivaani S**

**ABSTRACT**

A warehouse is place for storing goods. It is used by manufactures, importers, exporters, transport businesses, wholesalers, etc. In this recent technological world, warehouse management system is a software application used for the efficient management of warehouses.This project aims to develop a web application. This application is used to predict the demands required in a warehouse for a short period of time. Here, we aim to analyse the data regarding sales and production and to extract the daily data from warehouse which is used to detect the fluctuations in the sales. The web application uses machine learning algorithms to predict the requirements for production which helps in managing the warehouse. Then, we aim to deploy the created models using python.

**Table of Contents**

|  |  |
| --- | --- |
| **Contents** | **Page No.** |
| Abstract | i |
| Table of Contents | ii |

|  |  |  |
| --- | --- | --- |
| **Chapter 1:** | **Introduction** |  |
| **Chapter 2:** | **Objectives** |  |
| **Chapter 3:** | **System Design** |  |
| **Chapter 4:** | **System Implementation** |  |
| **Chapter 5:** | **Applications** |  |
| **Chapter 6:** | **Conclusion and Future Scope** |  |
|  | **References** |  |

**INTRODUCTION:**

A food delivery service has to deal with a lot of perishable raw materials which makes it all, the most important factor for such a company is to accurately forecast daily and weekly demand. Too much inventory in the warehouse means more risk of wastage, and not enough could lead to out-of-stocks - and push customers to seek solutions from your competitors. The replenishment of majority of raw materials is done on weekly basis and since the raw material is perishable, the procurement planning is of utmost importance.

**OBJECTIVES:**

Overloaded inventory and a customer leaving the store disappointed due to no stock of goods are some of the major problems in warehouse management. The inventory capacity must be aligned with the demand of customers in a given time frame. Hence, we propose the solution of developing a mobile and a web application to interact with the machine learning model to predict the demand for goods based on the fluctuating customer needs and supply situations. This system will help to avoid food wastage and provide a value added tool to improve the business revenue.

**SYSTEM DESIGN:**

Clustering and association algorithms were used to classify the users and the products based on their similarities. Feature scaling was used to standardize the range of independent variables of data. Outliers detection was done using Turkey’s Method. Principal Component Analysis was used to draw conclusions based on maximum variance calculations. Graphs were plotted and percentile ranks were taken and the demand for goods was calculated.The web application was built using the streamlit framework, and the developed using the android studio.

import numpy as np

import pandas as pd

from IPython.display import display

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeRegressor

from sklearn.decomposition import PCA

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib.cm as cm

import pandas as pd

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

data=pd.read\_csv (r'C:\Users\MAHIMA MANIGANDAN\Desktop\Hack\Dataset.csv')

print (data)

try:

    data = pd.read\_csv(r'C:\Users\MAHIMA MANIGANDAN\Desktop\Hack\Dataset.csv')

    data.drop(['Region', 'Date'], axis = 1, inplace = True)

    print("The dataset has {} samples with {} features each.".format(\*data.shape))

except:

    print("Dataset could not be loaded. Is the dataset missing?")

import warnings

warnings.filterwarnings("ignore", category = UserWarning, module = "matplotlib")

from IPython import get\_ipython

def pca\_results(good\_data, pca):

    dimensions= ['Dimension {}'.format(i) for i in range(1,len(pca.components\_)+1)]

    components= pd.DataFrame(np.round(pca.components\_, 4), columns = list(good\_data.keys()))

    components.index = dimensions

    ratios = pca.explained\_variance\_ratio\_.reshape(len(pca.components\_), 1)

    variance\_ratios = pd.DataFrame(np.round(ratios, 4), columns = ['Explained Variance'])

    variance\_ratios.index = dimensions

    fig, ax = plt.subplots(figsize = (14,8))

    components.plot(ax = ax, kind = 'bar')

    ax.set\_ylabel("Feature Weights")

    ax.set\_xticklabels(dimensions, rotation=0)

    for i, ev in enumerate(pca.explained\_variance\_ratio\_):

     ax.text(i-0.40, ax.get\_ylim()[1] + 0.05, "Explained Variance\n%.4f"%(ev))

     predictions = pd.DataFrame(preds, columns = ['Cluster'])

     plot\_data = pd.concat([predictions, reduced\_data], axis = 1)

     return pd.concat([variance\_ratios, components], axis = 1)

def cluster\_results(reduced\_data, preds, centers, pca\_samples):

    fig, ax = plt.subplots(figsize = (14,8))

    cmap = cm.get\_cmap('gist\_rainbow')

    for i, cluster in plot\_data.groupby('Cluster'):

        cluster.plot(ax = ax, kind = 'scatter', x = 'Dimension 1', y = 'Dimension 2', color = cmap((i)\*1.0/(len(centers)-1)), label = 'Cluster %i'%(i), s=30);

    for i, c in enumerate(centers):

        ax.scatter(x = c[0], y = c[1], color = 'white', edgecolors = 'black', alpha = 1, linewidth = 2, marker = 'o', s=200)

        ax.scatter(x = c[0], y = c[1], marker='$%d$'%(i), alpha = 1, s=100)

        ax.scatter(x = pca\_samples[:,0], y = pca\_samples[:,1], s = 150, linewidth = 4, color = 'black', marker = 'x')

        ax.set\_title("Cluster Learning on PCA-Reduced Data - Centroids Marked by Number\nTransformed Sample Data Marked by Black Cross")

def biplot(good\_data, reduced\_data, pca):

    fig, ax = plt.subplots(figsize = (14,8))

    ax.scatter(x=reduced\_data.loc[:, 'Dimension 1'], y=reduced\_data.loc[:, 'Dimension 2'],

    facecolors='b', edgecolors='b', s=70, alpha=0.5)

    feature\_vectors = pca.components\_.T

    arrow\_size, text\_pos = 7.0, 8.0,

    for i, v in enumerate(feature\_vectors):

     ax.arrow(0, 0, arrow\_size\*v[0], arrow\_size\*v[1], head\_width=0.2, head\_length=0.2, linewidth=2, color='red')

     ax.text(v[0]\*text\_pos, v[1]\*text\_pos, good\_data.columns[i], color='black', ha='center', va='center', fontsize=18)

     ax.set\_xlabel("Dimension 1", fontsize=14)

     ax.set\_ylabel("Dimension 2", fontsize=14)

     ax.set\_title("PC plane with original feature projections.", fontsize=16)

     return ax

def channel\_results(reduced\_data, outliers, pca\_samples):

    try:

        full\_data = pd.read\_csv("../input/customers.csv")

    except:

        print("Dataset could not be loaded. Is the file missing?")

        return False

    # Create the Channel DataFrame

    channel = pd.DataFrame(full\_data['Date'], columns = ['Date'])

    channel = channel.drop(channel.index[outliers]).reset\_index(drop = True)

    labeled = pd.concat([reduced\_data, channel], axis = 1)

    # Generate the cluster plot

    fig, ax = plt.subplots(figsize = (14,8))

    # Color map

    cmap = cm.get\_cmap('gist\_rainbow')

    # Color the points based on assigned Channel

    labels = ['Hotel/Restaurant/Cafe', 'Retailer']

    grouped = labeled.groupby('Date')

    for i, channel in grouped:

        channel.plot(ax = ax, kind = 'scatter', x = 'Dimension 1', y = 'Dimension 2', \

                     color = cmap((i-1)\*1.0/2), label = labels[i-1], s=30);

    # Plot transformed sample points

    for i, sample in enumerate(pca\_samples):

        ax.scatter(x = sample[0], y = sample[1], \

               s = 200, linewidth = 3, color = 'black', marker = 'o', facecolors = 'none');

        ax.scatter(x = sample[0]+0.25, y = sample[1]+0.3, marker='$%d$'%(i), alpha = 1, s=125);

    # Set plot title

    ax.set\_title("PCA-Reduced Data Labeled by 'Date'\nTransformed Sample Data Circled");

display(data.head())

display(data.info())

display(data.describe())

np.random.seed(2018)

indices = np.random.randint(low = 0, high = 441, size = 3)

print("Indices of Samples => {}".format(indices))

samples = pd.DataFrame(data.loc[indices], columns = data.keys()).reset\_index(drop = True)

print("\nChosen samples of the dataset:")

display(samples)

def sampl\_pop\_plotting(sample):

    fig, ax = plt.subplots(figsize=(10,5))

    index = np.arange(sample.count())

    bar\_width = 0.3

    opacity\_pop = 1

    opacity\_sample = 0.3

    rect1 = ax.bar(index, data.mean(), bar\_width,

                    alpha=opacity\_pop, color='g',

                    label='Population Mean')

    rect2 = ax.bar(index + bar\_width, sample, bar\_width,

                    alpha=opacity\_sample, color='k',

                    label='Sample')

    ax.set\_xlabel('Categories')

    ax.set\_ylabel('Total Purchase Cost')

    ax.set\_title('Sample vs Population Mean')

    ax.set\_xticks(index + bar\_width / 2)

    ax.set\_xticklabels(samples.columns)

    ax.legend(loc=0, prop={'size': 15})

    fig.tight\_layout()

    plt.show()

display(samples.iloc[0] - data.mean())

sampl\_pop\_plotting(samples.iloc[0])

display(samples.iloc[1] - data.mean())

sampl\_pop\_plotting(samples.iloc[1])

display(samples.iloc[2] - data.mean())

sampl\_pop\_plotting(samples.iloc[2])

percentiles\_data = 100\*data.rank(pct=True)

percentiles\_samples = percentiles\_data.iloc[indices]

plt.subplots(figsize=(10,5))

\_ = sns.heatmap(percentiles\_samples, annot=True)

def predict\_one\_feature(dropped\_feature):

    print("Dropping feature -> {}".format(dropped\_feature))

    new\_data = data.drop([dropped\_feature], axis = 1)

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(new\_data, data[dropped\_feature], test\_size=0.25, random\_state=0)

    regressor = DecisionTreeRegressor(random\_state=0)

    regressor.fit(X\_train, y\_train)

    score = regressor.score(X\_test, y\_test)

    print("Score for predicting '{}' using other features = {:.3f}\n".format(dropped\_feature, score))

    predict\_one\_feature('Milk')

    print("Features in data -> {}\n".format(data.columns.values))

    for cols in data.columns.values:

        predict\_one\_feature(cols)

corr = data.corr()

plt.figure(figsize = (10,5))

ax = sns.heatmap(corr, annot=True)

ax.legend(loc=0, prop={'size': 15})

for cols in data.columns.values:

    ax = sns.kdeplot(data[cols])

    ax.legend(loc=0, prop={'size': 15})

log\_data = np.log(data)

log\_samples = np.log(samples)

display(log\_samples)

log\_corr = log\_data.corr()

f = plt.figure(figsize = (16,8))

mask = np.zeros\_like(corr)

mask[np.triu\_indices\_from(mask)] = True

with sns.axes\_style("white"):

    ax1 = sns.heatmap(corr, annot=True, mask=mask, cbar\_kws={'label': 'Before Log Normalization'})

mask2 = np.zeros\_like(corr)

mask2[np.tril\_indices\_from(mask2)] = True

with sns.axes\_style("white"):

    ax2 = sns.heatmap(log\_corr, annot=True, mask=mask2, cmap="YlGnBu", cbar\_kws={'label': 'After Log Normalization'})

outliers\_list = []

for feature in log\_data.keys():

    Q1 = np.percentile(log\_data[feature], 25)

    Q3 = np.percentile(log\_data[feature], 75)

    step = (Q3 - Q1) \* 1.5

    print("Data points considered outliers for the feature '{}':".format(feature))

    outliers = list(log\_data[~((log\_data[feature] >= Q1 - step) & (log\_data[feature] <= Q3 + step))].index.values)

    display(log\_data[~((log\_data[feature] >= Q1 - step) & (log\_data[feature] <= Q3 + step))])

    outliers\_list.extend(outliers)

print("List of Outliers -> {}".format(outliers\_list))

duplicate\_outliers\_list = list(set([x for x in outliers\_list if outliers\_list.count(x) >= 2]))

duplicate\_outliers\_list.sort()

print("\nList of Common Outliers -> {}".format(duplicate\_outliers\_list))

outliers  = duplicate\_outliers\_list

good\_data = log\_data.drop(log\_data.index[outliers]).reset\_index(drop = True)

pca = PCA(n\_components = 6, random\_state=0)

pca.fit(good\_data)

pca\_samples = pca.transform(log\_samples)

print("Explained Variance Ratio => {}\n".format(pca.explained\_variance\_ratio\_))

print("Explained Variance Ratio(csum) => {}\n".format(pca.explained\_variance\_ratio\_.cumsum()))

pca = PCA(n\_components = 2, random\_state=0)

pca.fit(good\_data)

reduced\_data = pca.transform(good\_data)

pca\_samples = pca.transform(log\_samples)

reduced\_data = pd.DataFrame(reduced\_data, columns = ['Dimension 1', 'Dimension 2'])

display(pd.DataFrame(np.round(pca\_samples, 4), columns = ['Dimension 1', 'Dimension 2']))

biplot(good\_data, reduced\_data, pca)

def sil\_coeff(no\_clusters):

    clusterer\_1 = KMeans(n\_clusters=no\_clusters, random\_state=0 )

    clusterer\_1.fit(reduced\_data)

    preds\_1 = clusterer\_1.predict(reduced\_data)

    centers\_1 = clusterer\_1.cluster\_centers\_

    sample\_preds\_1 = clusterer\_1.predict(pca\_samples)

    score = silhouette\_score(reduced\_data, preds\_1)

    print("silhouette coefficient for `{}` clusters => {:.4f}".format(no\_clusters, score))

clusters\_range = range(2,15)

for i in clusters\_range:

    sil\_coeff(i)

clusterer = KMeans(n\_clusters = 2)

clusterer.fit(reduced\_data)

preds = clusterer.predict(reduced\_data)

centers = clusterer.cluster\_centers\_

sample\_preds = clusterer.predict(pca\_samples)

log\_centers = pca.inverse\_transform(centers)

true\_centers = np.exp(log\_centers)

segments = ['Segment {}'.format(i) for i in range(0,len(centers))]

true\_centers = pd.DataFrame(np.round(true\_centers), columns = data.keys())

true\_centers.index = segments

display(true\_centers)

display(data.mean(axis=0))

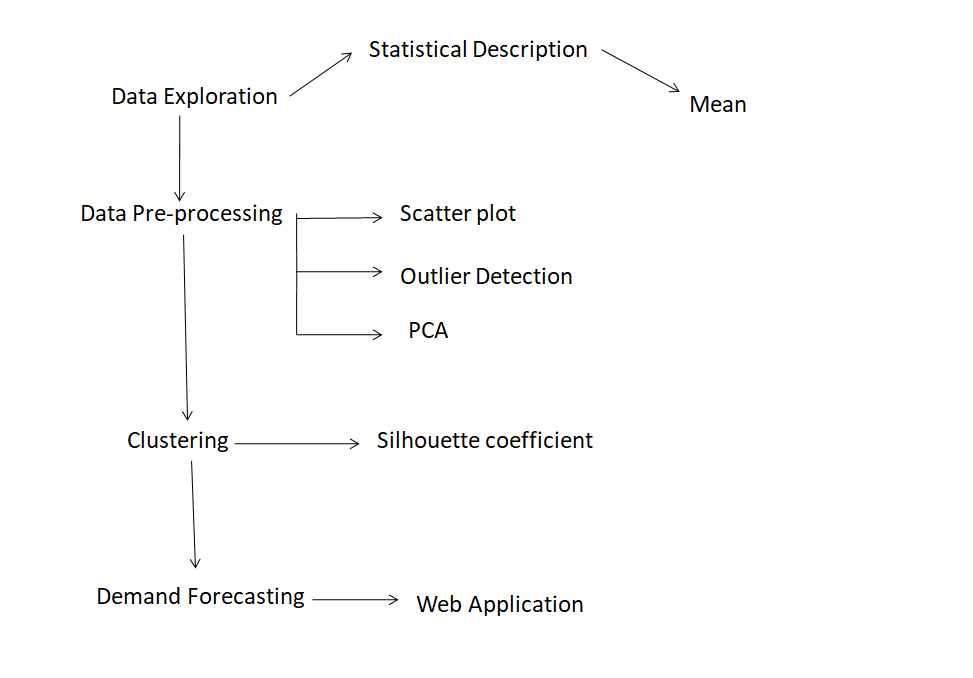
display(samples)

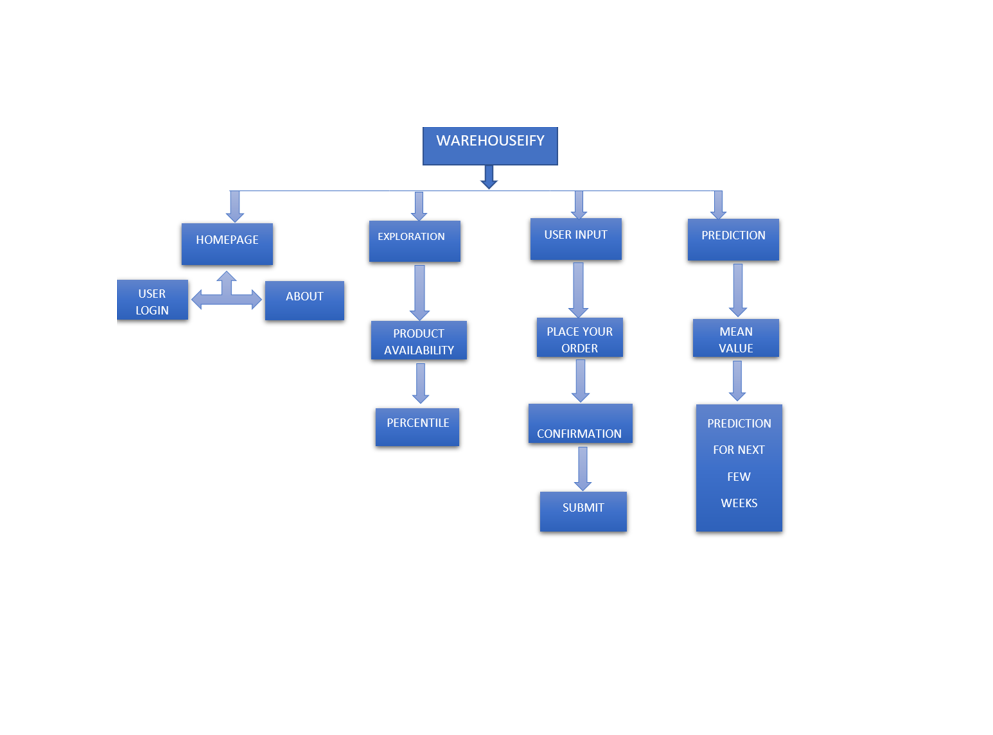
for i, pred in enumerate(sample\_preds):

    print("Sample point", i, "predicted to be in Cluster", pred)

channel\_results(reduced\_data, outliers, pca\_samples)

**SYSTEM IMPLEMENTATION:**

****

****

**APPLICATION:**

The demand requirements of the warehouse, for a short period of time, can be predicted. Since, we aim to predict the demand fluctuations; it saves the losses created by wastage of the perishable goods. We can also predict the effects if any delivery changes are made. As we have used association algorithm, it helps us to predict what the customer's frequently purchased together. This helps us to increase the sales as we can provide all the products that are frequently bought together. This demand forecasting concept can be applied in various areas of the supply chain, in order to prevent the wastage of the perishable goods. These predictions can be used to make any changes in the delivery service too. The web application serves as a personal advisor to predict the demand of the perishable food.

**CONCLUSION AND FUTURE SCOPE:**

In the past years, the efficiency of warehouse management has become an area of major concern in business. New models for managing the inventory levels are now available. Most of the analytical models addressed only one type of uncertainty and assumed a simple structure of the production process. The most common dimensions to be considered as variables are demand, the cost of acquisition. Each model, based on some assumptions, has its benefits and disadvantages. The existence of such quantity of models shows that machine learning models are one of the appropriate methods, which can suppose a great advance in inventory management. A warehouse has to deal with a lot of perishable food supply, which in case exceeds the daily or weekly demand can lead to wastage of resources. This wastage of the perishable goods can be greatly reduced when we predict the demand requirements on daily or weekly basis and accordingly maintain the stock in the warehouse. By following these predictions, sales can be maintained at a higher rate since; we are never in shortage of the goods requirement.

**REFERNCES:**

* M.I.M. Wahab, S.M.H. Mamun, P OngkunarukEOQ models for a coordinated two-level international supply chain considering imperfect items and environmental impact Int J Prod Econ, 134 (2011), pp. 151-158
* Understanding challenges to food security in dry arab micro-states: evidence from qatari micro-data NY: Social Science Research Network, Rochester (2013)
* Sustainable food security futures: perspectives on food waste and information across the food supply chain J Enterp Inf Manag, 29 (2016), 10.1108/JEIM-12-2015-0117171–8
* K.P MurphyMachine learning: a probabilistic perspective. edición: 1 MIT Press Ltd, Cambridge, MA (2012)
* G. James, D. Witten, T. Hastie, R TibshiraniAn introduction to statistical learning: with applications R. Edición (Ed.) (1st ed), Springer (2013) 2013, Corr. 6th printing 2016