1. **INTRODUCTION:**

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.

1.1 OVERVIEW:

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.

1.2 PURPOSE:

We aim to propose the demand of goods required for a warehouse, for the next few weeks. This is done by analysing the sales and production data of the warehouse. A web application is developed, which enables us to view the demand requirements.

2. LITERATURE SURVEY:

2.1 EXISTING PROBLEM:

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.

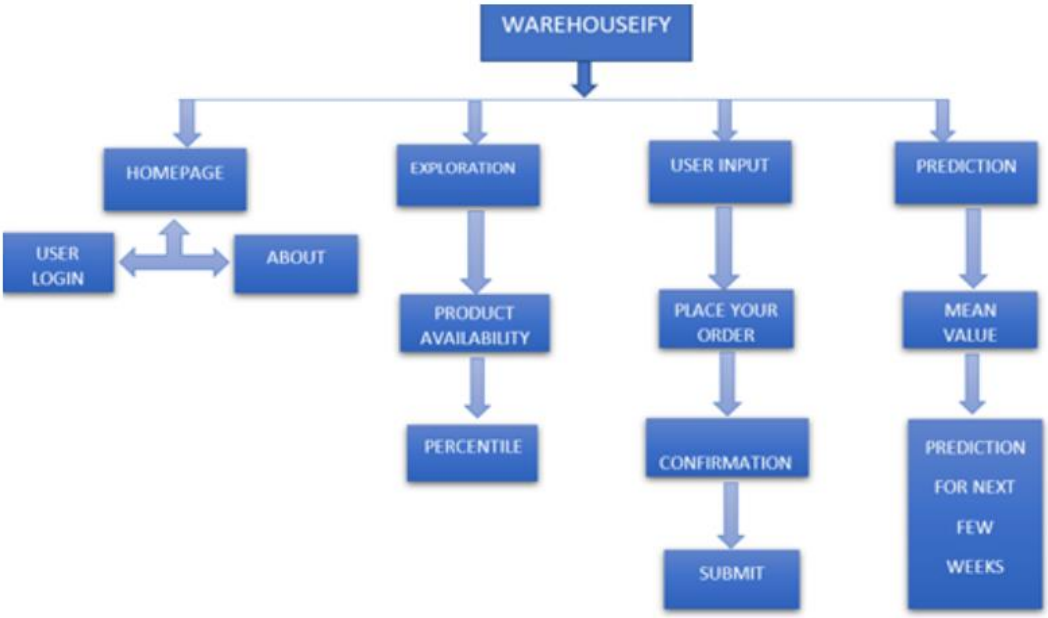
2.2 PROPOSED SOLUTION:

Machine Learning Model to predict the demand of goods for the next few weeks.

3. THEORITICAL ANALYSIS:

Predicting the demand requirements for a warehouse is an essential component for its management. Analyzing the production and sales transactions helps us to define a better marketing stratergy. Data analysis deals with collection and analysis of the dataset. The dataset was taken from the kaggle website. Clustring and Association algorithms were used to classify the data based on their similarities. Feature scaling was used to standardize the range of independant variables of data. Outliers detection was done using Turkey's Method. Principle Component Analysis(PCA) was used to draw conclusions based on the maximum variance calculations. Graphs were plotted and percentile ranks of the goods were taken and the demand of the goods wes calculated.

3.1 BLOCK DIAGRAM:



4. SOFTWARE DESIGNING:

```
import streamlit as st

import numpy as np
```

```
import pandas as pd

import matplotlib.cm as cm

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.tree import DecisionTreeRegressor

from sklearn.decomposition import PCA

from sklearn.model_selection import train_test_split

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette_score




def main():



    df=pd.read_csv(r'C:\Users\Shivaani\Desktop\Wholesale
   customers data.csv')

    page = st.sidebar.selectbox("Choose a page",
   ['Homepage', 'Exploration', 'User Input','Prediction'])

if page == 'Login':

      st.title('Enter Your Details')

      t1 = st.text_input("Email")



      st.sidebar.markdown("---")



      t2 = st.text_input("Password")



      st.write(t1)

      st.write(t2)
```

```
    button = st.button("Submit")




  elif page == 'About':

    st.title('Warehouseify')

    st.markdown(f'<div class="markdown-text-container
stText" style="width: 698px;"><div style="font-size:
medium;">Most products lose their market value (outdate)
over time. Some products lose valuefaster than others;
these are known as perishable products. Traditionally,
perishables outdate due to their chemical structure.
Examples of such perishable products are grocery, fresh
produce, frozen products, dairy products, delicassens
etc. So  This application is used to predict the demands
required in a warehouse for a short period of time. So,
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. </div>',unsafe_allow_html=True)



  elif page == 'Exploration':

    st.title('Explore the Product Availability')

    if st.checkbox('Show column descriptions'):

      st.dataframe(df.describe())

    st.markdown('### Analysing column relations')

    st.text('Correlations:')

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

    sns.heatmap(df.corr(), annot=True, ax=ax)

    st.pyplot()

    st.text('Effect of the different classes')
```

```
    sns.pairplot(df, vars=['Fresh', 'Milk', 'Grocery',
'Frozen', 'Detergents_Paper', 'Delicassen'],
hue='Channel')

    st.pyplot()

    st.line_chart(df)

  elif page == 'User Input':

      st.header('Place your order')

      Fresh = st.slider('Fresh', 3, 56000, 28001)

      Milk = st.slider('Milk', 50, 80000, 40025)

      Grocery = st.slider('Grocery', 3, 92780, 46390)

      Frozen = st.slider('Frozen', 25, 63869, 31947)

      Detergents_Paper = st.slider('Detergents_Paper', 3,
40827, 20415)

      Delicassen = st.slider('Delicassen', 3, 47943,
24000)

      warehouse = {'Fresh': Fresh,

                   'Milk': Milk,

                   'Grocery': Grocery,

                   'Frozen': Frozen,

                   'Detergents_Paper': Detergents_Paper,

                   'Delicassen': Delicassen}

      features = pd.DataFrame(warehouse, index=[0])

      st.write("""

        *You selected*""",warehouse)



      button = st.button("Submit")

  else:
```

```
      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')

      def channel_results(reduced_data, outliers,
pca_samples):

         try:
```

```
            full_data = 
pd.read_csv(r'C:\Users\Shivaani\Desktop\Wholesale 
customers data.csv')

          except:

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

             return False

          channel = pd.DataFrame(full_data['Channel'], 
columns = ['Channel'])

          channel = 
channel.drop(channel.index[outliers]).reset_index(drop = 
True)

          labeled = pd.concat([reduced_data, channel], axis 
= 1)

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

          cmap = cm.get_cmap('gist_rainbow')

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

          grouped = labeled.groupby('Channel')

          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)

          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)

             ax.set_title("PCA-Reduced Data Labeled by 
'Channel'\nTransformed Sample Data Circled")

       st.write(df.describe())

       np.random.seed(2018)
```

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

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



      samples = pd.DataFrame(df.loc[indices], columns =
df.keys()).reset_index(drop = True)

      print("\nChosen samples of wholesale customers
dataset:")



      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 = df.corr()

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

    ax = sns.heatmap(corr, annot=True)

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

    for cols in df.columns.values:

        ax = sns.kdeplot(df[cols])

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

    log_data = np.log(df)

    log_samples = np.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)
```

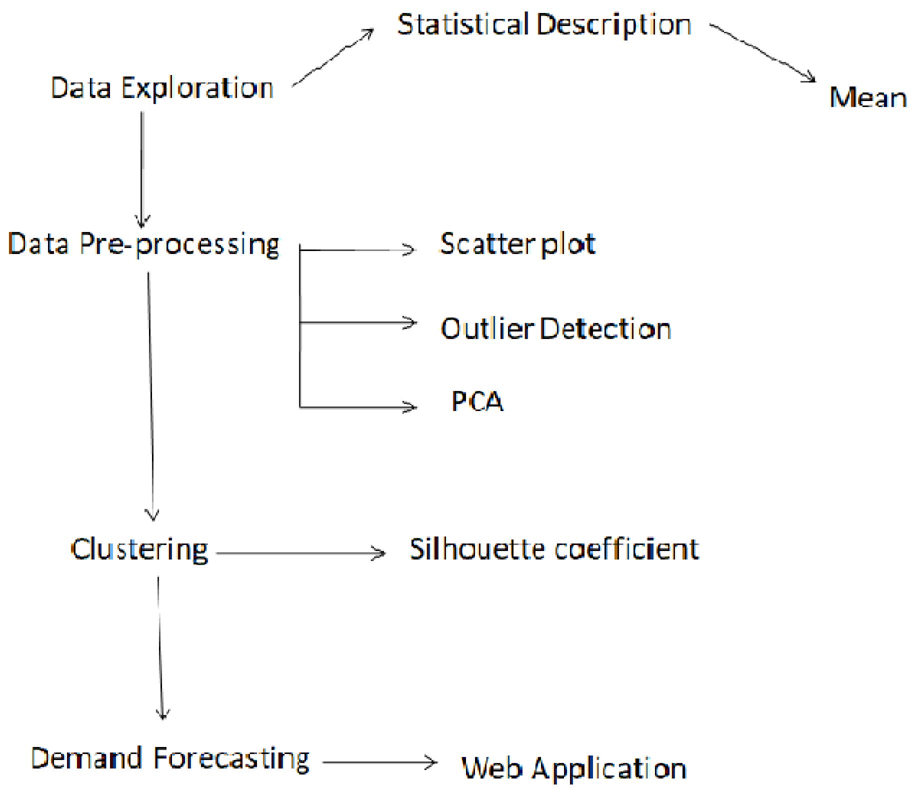
```
    mask2[np.tril_indices_from(mask2)] = True

    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)

        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'])



      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)
```

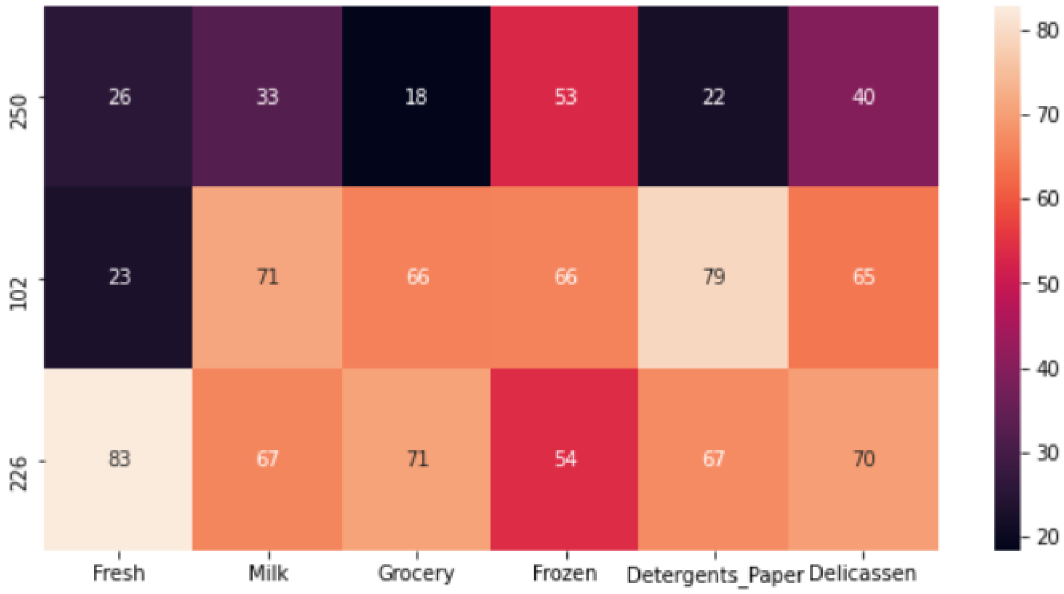
```
        clusterer = KMeans(n_clusters = 2)

        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 = df.keys())

        true_centers.index = segments

        st.write(samples)



        for i, pred in enumerate(sample_preds):

            st.write("Sample point", i, "predicted to be in
   Cluster", pred)

        channel_results(reduced_data, outliers, pca_samples)



@st.cache(allow_output_mutation=True)

def train_model(df):

    X = np.array(df.drop(['Milk'], axis=1))

    y= np.array(df['Milk'])



    X_train, X_test, y_train, y_test = train_test_split(X,
   y, test_size=0.2, random_state=42)



    model = RandomForestClassifier()
```

```
    model.fit(X_train, y_train)



    return model, model.score(X_test, y_test)



@st.cache

def load_data():

    return pd.read_csv(r'C:\Users\Shivaani\Desktop\Wholesale
customers data.csv')

    names=['Fresh', 'Milk', 'Grocery', 'Frozen',
'Detergents_Paper', 'Delicassen']



if __name__ =='__main__':

    main()
```

5. FLOWCHART:


Statistical Description
Data Exploration
Mean
Data Pre-processing
Scatter plot
Outlier Detection
PCA
Clustering
Silhouette coefficient
Demand Forecasting
Web Application

6. RESULT:



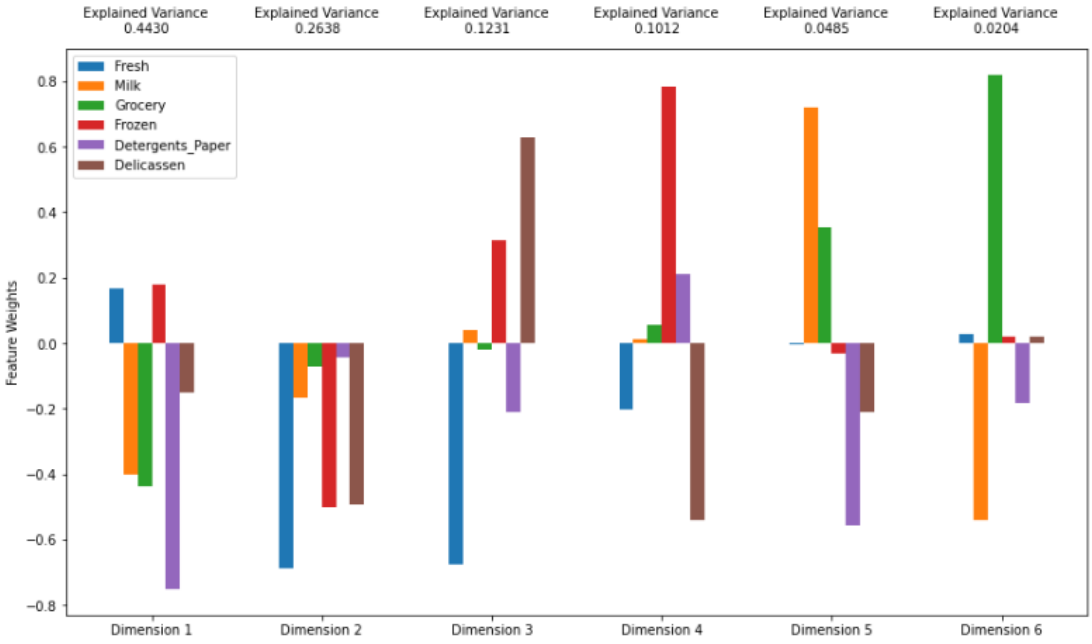
The output of the above command helps to draw conclusion for three samples:

➢ Sample 1: From the first sample, it can be concluded that the establishment can be of a smaller range, since the purchase cost of all the categories is less than the population mean.

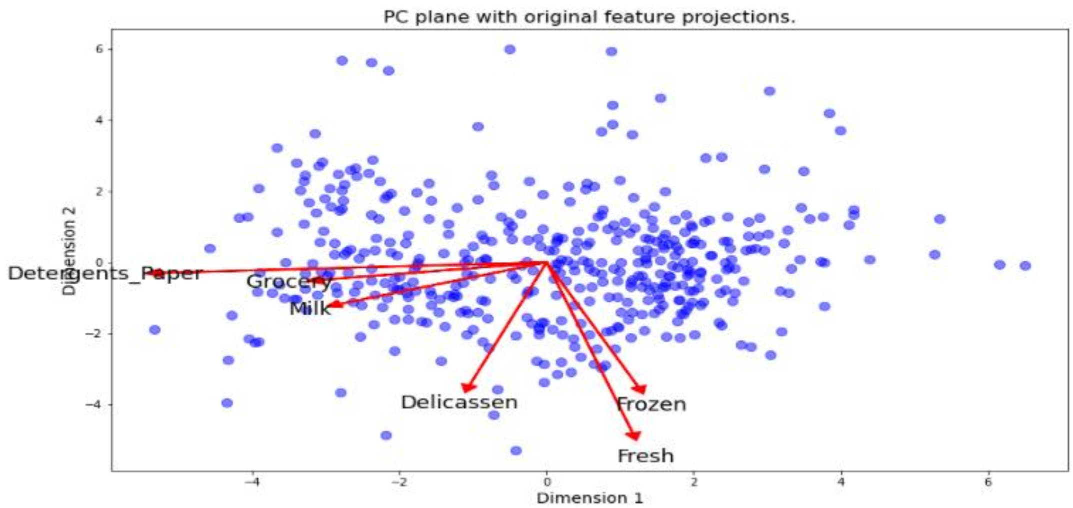
➢ Sample 2: From the next sample, it can be concluded that the establishment is a little bigger comperd to the first sample, since the purchase cost is approximately equal to the population mean of Milk, Grocery and Frozen.

➢ Sample 3: From the last sample, it can be concluded that it shows the data for a retailer, since the purchase cost of all the categories is equal to the population mean of all the categories except for Fresh, which shows a high purchase cost.

```
Explained Variance Ratio => [0.44302505 0.26379218 0.1230638  0.10120908 0.04850196 0.02040793]

Explained Variance Ratio(csum) => [0.44302505 0.70681723 0.82988103 0.93109011 0.97959207
1.        ]
```



The total variance in the data exhibited by the first and second principal component is found to be 70.68% (0.4430 + 0.2637 = 0.7068) and the variance between the first four principal components (93.11%).



From the biplot, once the original feature projections (those shown in red) are identified, it is easier to interpret the relative position of each data in the scatter plot. For instance, a point the lower right corner of the figure will likely correspond to a customer that spends a lot on Milk, Grocery and Detergents_Paper, but not so much on the other product categories.

```
silhouette coefficient for `2` clusters => 0.4263
silhouette coefficient for `3` clusters => 0.3974
silhouette coefficient for `4` clusters => 0.3312
silhouette coefficient for `5` clusters => 0.3510
silhouette coefficient for `6` clusters => 0.3637
silhouette coefficient for `7` clusters => 0.3649
silhouette coefficient for `8` clusters => 0.3663
silhouette coefficient for `9` clusters => 0.3633
silhouette coefficient for `10` clusters => 0.3496
silhouette coefficient for `11` clusters => 0.3616
silhouette coefficient for `12` clusters => 0.3548
silhouette coefficient for `13` clusters => 0.3655
silhouette coefficient for `14` clusters => 0.3603
```

From the above output, we find the the silhouette coefficient of 2 clusters to be the highest.

➢ silhouette coefficient of 2 cluster: 0.4263

➢ silhouette coefficient of 3 cluster: 0.3969

From the given dataset, the following assumptions can be made:

➢ Segment 0 customers are purchasing more from the Fresh and Frozen products than the segment 1 customers.

➢ Segment 1 customers are purchasing more from the Detergents_Paper and Delicassen products than the segment 0 customers.

From each data point, the following predictions can be made:

➢ Sample 0: This customer purchased very less of Milk, Grocery, Detergents_Paper and definitely falls in Segment 0.

➢ Both Sample 1 and 2 : These customers purchased a lot of 'Milk', 'Grocery', 'Detergents_Paper' and falls in Segment 1.

Segment 1 customers are spending more on Frozen and Fresh and obviously they will love to get the delivery every day. So, if we change the delivery service from 5 days a week to 3 days a week, Segment 1 customers will react very badly.

If the wholesale distributor is considering changing its delivery service from currently 5 days a week to 3 days a week then he should first test it on Segment 0 customers and see their reaction first before he can try it out for Segment 1 customers.

For the A/B testing, the wholesale distributor should select random samples that significantly summarizes the population from Segment 0 and Segment 1. Now, the sample has to be split in to 2 groups where one of the group will act as a control group. Now, the wholesale distributor should change the delivery service from 5 days a week to 3 days a week and check what segment of people have a low satisfaction rate. This A/B testing will help the wholesale distributor to change the delivery service from 5 days a week to 3 days a week for only those segment of customers who reacted positively or who did not react negatively.

7. ADVANTAGES AND DISADVANTAGES:

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 looses 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 hepls 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 entire prediction may very from time to time. It entirely depends on the customers present in the region and also on the population mean of the surrounding area.

8. APPLICATIONS:

This demand forecasting concept can be applied in various areas of the supply chain, in order to prevent the wastage of the perishable goods. This 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.

9. CONCLUSION:

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.

10. FUTURE SCOPE:

A warehouse has to deal with a lot of perishable food supply, which in case exceeds the daily or weekly demand can lead to a wastage of resources. This wastage of the perishable goods can be greately reduced when we predict the demand requirements on daily or weekly basis and accordingly maintain the stock in the warehouse. By following this predictions, sales can be maintained at a higher rate since, we are never in shortage of the goods requirement.

11. BIBLIOGRAPHY:

➢ M.I.M. Wahab, S.M.H. Mamun, P Ongkunaruk**EOQ 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 Murphy**Machine learning: a probabilistic perspective. edición: 1**

MIT Press Ltd, Cambridge, MA (2012)

➢ G. James, D. Witten, T. Hastie, R Tibshirani**An introduction to statistical learning: with applications**

R. Edición (Ed.) (1st ed), Springer (2013)

2013, Corr. 6th printing 2016