PREDIKSI PENJUALAN WALMART MENGGUNAKAN MACHINE LEARNING ALGORITHMS.

Presentation by Kelompok 3

INTRODUCTION

Nama Kelompok :
1. Khadafy Rafsanjani (JumaTec)
2. Farah Tatania (JumaTec)
3. Ayu Susanti (JumaTec)
4. Nurul Syafdiah (Winning Eleven)
5. Flenco Nerohoston (Winning Eleven)
6. Ami widyas (jumatec)
7. Risma Mardiantika P (Winning Eleven)

WORKFLOW

- Business Concept
- Tujuan
- Manfaat
- Data Understanding
- Data Exploration
- Explotary Data Analysis (EDA)
- Data Pre-Processing
- Data Manipulation
- Feature Selection/Extraction
- Predictive Modelling
- Conclusion

BUSINESS CONCEPT

Walmart adalah perusahaan Amerika Serikat yang mengelola jaringan toserba. Menurut Fortune Global 500 2008, Walmart adalah perusahaan terbuka terbesar di dunia berdasarkan pendapatan. Bisnis ini menghadapi tantangan karena permintaan yang tidak terduga dan kehabisan stok beberapa kali, karena algoritma pembelajaran mesin yang tidak tepat. Oleh karena itu dibutuhkan alat untuk memprediksi penjualan dan permintaan secara akurat. Ada banyak algoritma pembelajaran mesin yang tepat, seperti Algoritma ML yang ideal akan memprediksi permintaan secara akurat menyerap faktor-faktor seperti kondisi ekonomi termasuk CPI, indeks pengangguran, dll. Prediksi penjualan atau peramalan adalah metode untuk memperkirakan suatu nilai dimasa dengan menggunakan data masa lalu.

TUJUAN

01 Memprediksi penjualan dan permintaan barang secara akurat.

02 Memperkirakan penjualan yang akurat memungkinkan perusahaan membuat keputusan bisnis yang tepat.

03 Memprediksi kinerja jangka pendek dan jangka panjang.

MANFAAT

Dengan memanfaatkan sales prediktor kita bisa memberikan informasi kepada Walmart itu tentang faktor-faktor yang mempengaruhi penjualan. Sehingga kita dapat menghadapi tantangan permintaan barang yang tidak terduga serta kehabisan stock.

DATA UNDERSTANDING

- Store
- Date
- Weekly sales
- Holiday Flag - whether the week is a special holiday week 1 - holiday week 0 - non holiday week
- Temperature
- Fuel Price
- CPI
- Unemployment

DATA EXPLORATION

In [2]:

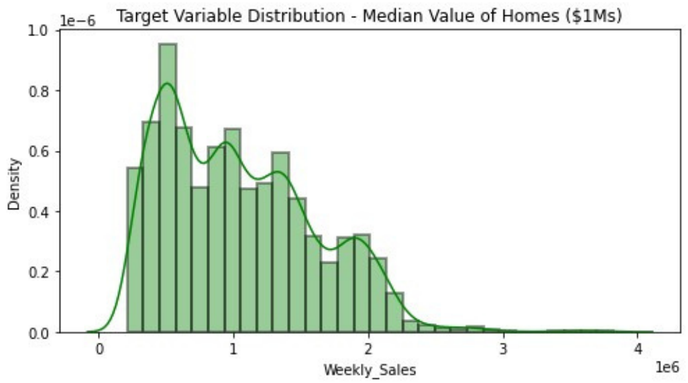
```
#Importing the basic libraries

import os
import math
import numpy as np
import pandas as pd
import seaborn as sns
from IPython.display import display

#from brokenaxes import brokenaxes
from statsmodels.formula import api
from sklearn.feature_selection import RFE
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn.decomposition import PCA
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = [10,6]

import warnings
warnings.filterwarnings('ignore')
```

```
#Importing the dataset

df = pd.read_csv('../input/walmart-dataset/Walmart.csv')

#df.drop(['car name'], axis=1, inplace=True)
display(df.head())

original_df = df.copy(deep=True)

print('\n\033[1mInference:\033[0m The Datset consists of {} features & {} samples.'.format(df.shape[1], df.shape[0]))
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106

EXPLORATORY DATA ANALYSIS (EDA)

```
#Let us first analyze the distribution of the target variable

plt.figure(figsize=[8,4])
sns.distplot(df[target], color='g',hist_kws=dict(edgecolor="black", linewidth=2), bins=30)
plt.title('Target Variable Distribution - Median Value of Homes ($1Ms)')
plt.show()
```



Tujuan utama Exploratory Data Analysis (EDA) adalah untuk membantu melihat data sebelum membuat asumsi apa pun. Ini dapat membantu mengindentifikasi kesalahan yang jelas, serta lebih memahami pola dalam data, mendeteksi outlier atau peristiwa anomali, menemukan hubungan yang menarik di antara variabel.

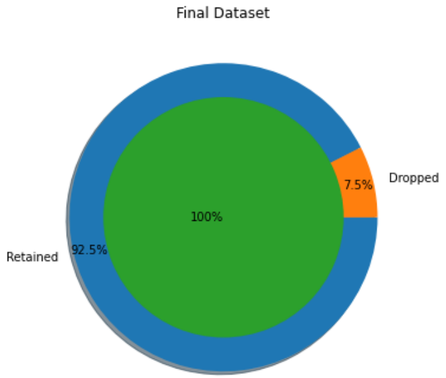
DATA PREPROCESSING

```
#Removal of outlier:

df1 = df3.copy()

#features1 = [i for i in features if i not in ['CHAS','RAD']]
features1 = nf

for i in features1:
    Q1 = df1[i].quantile(0.25)
    Q3 = df1[i].quantile(0.75)
    IQR = Q3 - Q1
    df1 = df1[df1[i] <= (Q3+(1.5*IQR))]
    df1 = df1[df1[i] >= (Q1-(1.5*IQR))]
    df1 = df1.reset_index(drop=True)
display(df1.head())
print('\n\033[1mInference:\033[0m\nBefore removal of outliers, The dataset had {} samples.'.form
at(df3.shape[0]))
print('After removal of outliers, The dataset now has {} samples.'.format(df1.shape[0]))
```

	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	year_2011	year_2012	weekday_1
0	1643690.90	0	42.31	2.572	211.096358	8.106	0	0	0
1	1641957.44	1	38.51	2.548	211.242170	8.106	0	0	0
2	1611968.17	0	39.93	2.514	211.289143	8.106	0	0	0
3	1409727.59	0	46.63	2.561	211.319643	8.106	0	0	0
4	1554806.68	0	46.50	2.625	211.350143	8.106	0	0	0

DATA PREPROCESSING

```
#Final Dataset size after performing Preprocessing

df = df1.copy()
df.columns=[i.replace('-','_') for i in df.columns]

plt.title('Final Dataset')
plt.pie([df.shape[0], original_df.shape[0]-df.shape[0]], radius = 1, labels=['Retained','Droppe
d'], counterclock=False,
        autopct='%1.1f%%', pctdistance=0.9, explode=[0,0], shadow=True)
plt.pie([df.shape[0]], labels=['100%'], labeldistance=-0, radius=0.78)
plt.show()

print(f'\n\033[1mInference:\033[0m After the cleanup process, {original_df.shape[0]-df.shape[0]}
samples were dropped, \
while retaining {round(100 - (df.shape[0]*100/(original_df.shape[0])),2)}% of the data.')
```



DATA MANIPULATION

```
#Splitting the data intro training & testing sets

m=[]
for i in df.columns.values:
    m.append(i.replace(' ','_'))

df.columns = m
X = df.drop([target],axis=1)
Y = df[target]
Train_X, Test_X, Train_Y, Test_Y = train_test_split(X, Y, train_size=0.8, test_size=0.2, random_
state=100)
Train_X.reset_index(drop=True,inplace=True)

print('Original set  ---> ',X.shape,Y.shape,'\nTraining set  ---> ',Train_X.shape,Train_Y.shap
e,'\nTesting set   ---> ', Test_X.shape,'', Test_Y.shape)
```

```
Original set  --->  (5953, 68) (5953,)
Training set  --->  (4762, 68) (4762,)
Testing set   --->  (1191, 68)  (1191,)
```

DATA MANIPULATION

```
#Feature Scaling (Standardization)

std = StandardScaler()

print('\033[1mStandardardization on Training set'.center(120))
Train_X_std = std.fit_transform(Train_X)
Train_X_std = pd.DataFrame(Train_X_std, columns=X.columns)
display(Train_X_std.describe())

print('\n','\033[1mStandardardization on Testing set'.center(120))
Test_X_std = std.transform(Test_X)
Test_X_std = pd.DataFrame(Test_X_std, columns=X.columns)
display(Test_X_std.describe())
```

Standardardization on Training set

	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	year_2011	year_2012
count	4.762000e+03	4.762000e+03	4.762000e+03	4.762000e+03	4.762000e+03	4.762000e+03	4.762000e+03
mean	-1.741339e-16	-1.494674e-16	-3.367039e-16	-2.799804e-16	-4.039888e-16	3.964583e-16	3.839385e-16
std	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00	1.000105e+00
min	-2.742012e-01	-2.961575e+00	-1.871814e+00	-1.248731e+00	-2.762670e+00	-7.526270e-01	-6.371530e-01
25%	-2.742012e-01	-7.314248e-01	-9.886990e-01	-1.076949e+00	-6.783836e-01	-7.526270e-01	-6.371530e-01
50%	-2.742012e-01	1.062547e-01	1.663112e-01	3.842133e-01	9.596435e-02	-7.526270e-01	-6.371530e-01
75%	-2.742012e-01	7.731979e-01	8.427860e-01	9.933828e-01	6.138095e-01	1.328679e+00	1.569482e+00
max	3.646958e+00	2.170008e+00	2.469806e+00	1.340791e+00	2.575491e+00	1.328679e+00	1.569482e+00

8 rows × 68 columns

Standardardization on Testing set

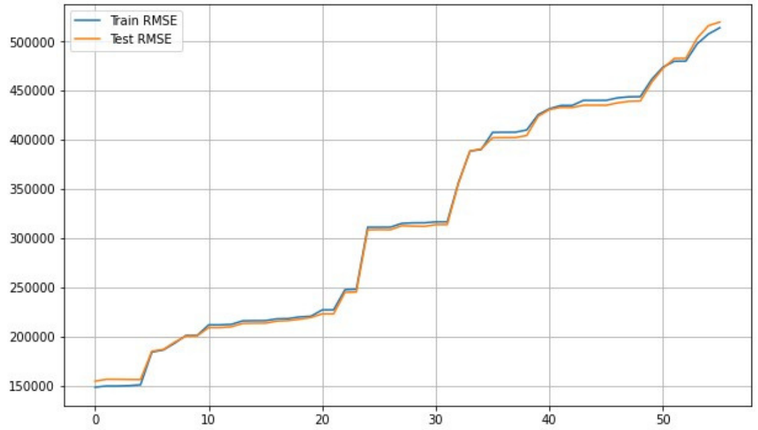
	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	year_2011	year_2012	weekday_1
count	1191.000000	1191.000000	1191.000000	1191.000000	1191.000000	1191.000000	1191.000000	1191.000
mean	0.005646	0.044406	0.075113	0.021041	-0.050953	0.052984	0.065042	0.007679
std	1.009885	1.000220	0.971917	1.004644	1.010206	1.014188	1.028250	1.014142
min	-0.274201	-2.857425	-1.780457	-1.248731	-2.762670	-0.752627	-0.637153	-0.25883
25%	-0.274201	-0.657516	-0.852751	-1.077025	-0.699355	-0.752627	-0.637153	-0.25883
50%	-0.274201	0.187351	0.298996	0.393492	0.058860	-0.752627	-0.637153	-0.25883
75%	-0.274201	0.818764	0.844961	1.019967	0.611390	1.328679	1.569482	-0.25883
max	3.646958	2.035481	2.469806	1.345814	2.575491	1.328679	1.569482	3.863473

FEATURE SELECTION/EXTRACTION

MANUAL METHOD - VARIANCE INFLATION FACTOR (VIF)

```
from sklearn.preprocessing import PolynomialFeatures
Trr=[]; Tss=[]; n=3
order=['ord-'+str(i) for i in range(2,n)]
#Trd = pd.DataFrame(np.zeros((10,n-2)), columns=order)
#Tsd = pd.DataFrame(np.zeros((10,n-2)), columns=order)

DROP=[];b=[]

for i in range(len(Train_X_std.columns)):
    vif = pd.DataFrame()
    X = Train_X_std.drop(DROP,axis=1)
    vif['Features'] = X.columns
    vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
    vif['VIF'] = round(vif['VIF'], 2)
    vif = vif.sort_values(by = "VIF", ascending = False)
    vif.reset_index(drop=True, inplace=True)
    if vif.loc[0][1]>1:
        DROP.append(vif.loc[0][0])
        LR = LinearRegression()
        LR.fit(Train_X_std.drop(DROP,axis=1), Train_Y)

        pred1 = LR.predict(Train_X_std.drop(DROP,axis=1))
        pred2 = LR.predict(Test_X_std.drop(DROP,axis=1))

        Trr.append(np.sqrt(mean_squared_error(Train_Y, pred1)))
        Tss.append(np.sqrt(mean_squared_error(Test_Y, pred2)))

        #Trd.loc[i,'ord-'+str(k)] = round(np.sqrt(mean_squared_error(Train_Y, pred1)),2)
        #Tsd.loc[i,'ord-'+str(k)] = round(np.sqrt(mean_squared_error(Test_Y, pred2)),2)

print('Dropped Features --> ',DROP)
#plt.plot(b)
#plt.show()
#print(API.summary())

# plt.figure(figsize=[20,4])
# plt.subplot(1,3,1)
# sns.heatmap(Trd.loc[:6], cmap='BuGn', annot=True, vmin=0, vmax=Trd.max().max())
# plt.title('Train RMSE')
# plt.subplot(1,3,2)
# sns.heatmap(Tsd.loc[:6], cmap='BuGn', annot=True, vmin=0, vmax=Trd.max().max()+10)
# plt.title('Test RMSE')
# plt.subplot(1,3,3)
# sns.heatmap((Trd+Tsd).loc[:6], cmap='BuGn', annot=True, vmin=0, vmax=Trd.max().max()+25)
# plt.title('Total RMSE')
# plt.show()

plt.plot(Trr, label='Train RMSE')
plt.plot(Tss, label='Test RMSE')
#plt.ylim([19.75,20.75])
plt.legend()
plt.grid()
plt.show()
```

```
Dropped Features -->  ['CPI', 'Unemployment', 'Fuel_Price', 'weekday_4', 'month_7', 'Store_
7', 'Temperature', 'month_12', 'Store_43', 'year_2012', 'Store_30', 'month_2', 'month_11', 'S
tore_16', 'month_5', 'Store_25', 'Store_29', 'month_10', 'Store_17', 'Holiday_Flag', 'Store_1
8', 'year_2011', 'Store_19', 'month_9', 'Store_20', 'Store_8', 'Store_34', 'Store_15', 'Store
_22', 'month_6', 'Store_21', 'Store_35', 'Store_14', 'Store_13', 'Store_45', 'Store_27', 'mon
th_3', 'weekday_1', 'Store_23', 'Store_44', 'Store_42', 'Store_11', 'weekday_5', 'Store_39',
'weekday_2', 'weekday_3', 'Store_24', 'Store_41', 'Store_40', 'Store_10', 'Store_36', 'Store_
9', 'month_4', 'Store_2', 'Store_3', 'Store_6']
```



PREDICTIVE MODELLING

POLYNOMIAL REGRESSION MODEL

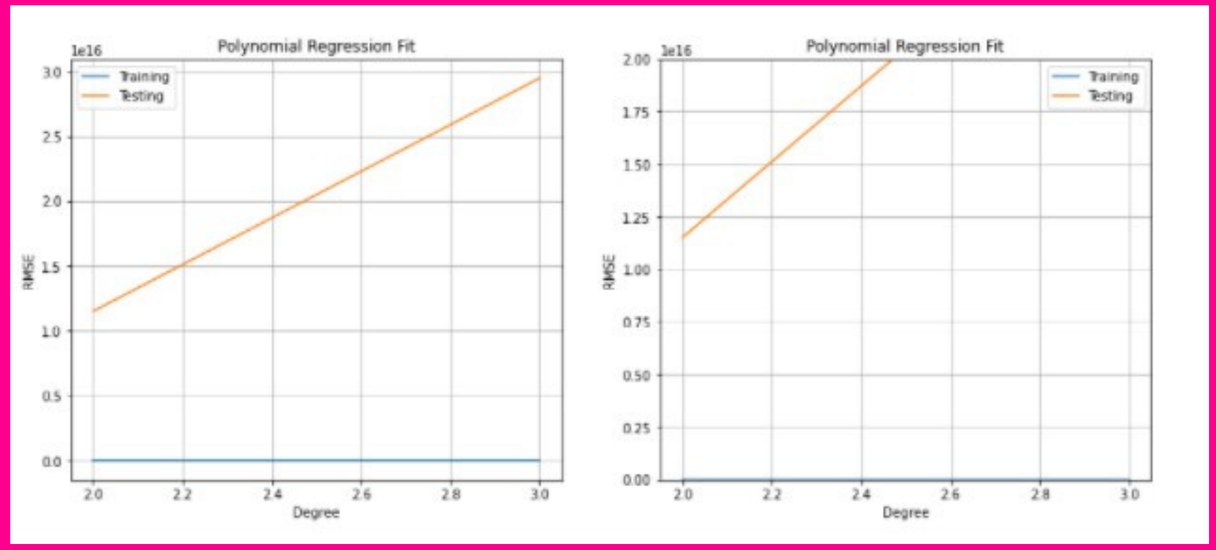
Polynomial Regression : Order-n

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

PREDICTIVE MODELLING

POLYNOMIAL REGRESSION MODEL

```
#Checking polynomial regression performance on various degrees

Trr=[]; Tss=[]
n_degree=4

for i in range(2,n_degree):
    #print(f'{i} Degree')
    poly_reg = PolynomialFeatures(degree=i)
    X_poly = poly_reg.fit_transform(Train_X_std)
    X_poly1 = poly_reg.fit_transform(Test_X_std)
    LR = LinearRegression()
    LR.fit(X_poly, Train_Y)

    pred1 = LR.predict(X_poly)
    Trr.append(np.sqrt(mean_squared_error(Train_Y, pred1)))

    pred2 = LR.predict(X_poly1)
    Tss.append(np.sqrt(mean_squared_error(Test_Y, pred2)))

plt.figure(figsize=[15,6])
plt.subplot(1,2,1)
plt.plot(range(2,n_degree),Trr, label='Training')
plt.plot(range(2,n_degree),Tss, label='Testing')
#plt.plot([1,4],[1,4],'b--')
plt.title('Polynomial Regression Fit')
#plt.ylim([0,5])
plt.xlabel('Degree')
plt.ylabel('RMSE')
plt.grid()
plt.legend()
#plt.xticks()

plt.subplot(1,2,2)
plt.plot(range(2,n_degree),Trr, label='Training')
plt.plot(range(2,n_degree),Tss, label='Testing')
plt.title('Polynomial Regression Fit')
plt.ylim([0,2e16])
plt.xlabel('Degree')
plt.ylabel('RMSE')
plt.grid()
plt.legend()
#plt.xticks()
plt.show()
```



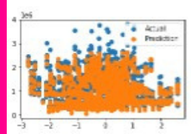
PREDICTIVE MODELLING

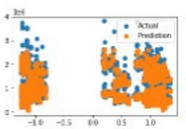
POLYNOMIAL REGRESSION MODEL

```
#Using the 2nd Order Polynomial Regression model (degree=2)

poly_reg = PolynomialFeatures(degree=2)
X_poly = poly_reg.fit_transform(Train_X_std)
X_poly1 = poly_reg.fit_transform(Test_X_std)
PR = LinearRegression()
PR.fit(X_poly, Train_Y)

pred1 = PR.predict(X_poly)
pred2 = PR.predict(X_poly1)

print('{}{}\033[1m Evaluating Polynomial Regression Model \033[0m{}{}\n'.format('<'*3,'-'*35
,'-'*35,'>'*3))
print('The Coeffecient of the Regresion Model was found to be ',MLR.coef_)
print('The Intercept of the Regresion Model was found to be ',MLR.intercept_)

Evaluate(4, pred1, pred2)
```

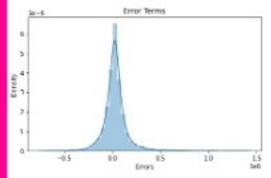
```
<<<---------------------------------- Evaluating Polynomial Regression Model --------------
```

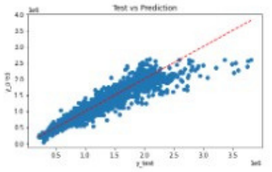
```
The Coeffecient of the Regresion Model was found to be  [ 382416.72707697  -87749.88819447
-33466.21811552  -70688.43596712
   23371.88280837   44272.12738693   67885.30758384 -171730.97447116
  194118.33861399 -182942.83065485  -95268.47858719 -118787.17299967
 -154257.07523525  280572.26744363  -27882.83358985   36827.35653929
  202356.39957846  137960.5727378  -126652.1156391    29819.45548371
   75864.89398192  119328.98106792  184372.22381835 -108321.57336784
   54083.53671998   65645.86638736  110781.93941655 -114111.68634244
   52174.80531825  169276.22877422   47325.81863416 -164944.63849199
  -45218.40228284   78308.2717848    42369.70843942 -163871.15661923
 -143446.53544316  -94113.42272716  -57253.08887135  -49965.33157082]
The Intercept of the Regresion Model was found to be  1847603.298112138
```






```
--------------------Training Set Metrics--------------------

R2-Score on Training set ---> 0.940476691273314
Residual Sum of Squares (RSS) on Training set  ---> 91733525612284.82
Mean Squared Error (MSE) on Training set       ---> 19263655185.477535
Root Mean Squared Error (RMSE) on Training set ---> 138793.57811575694

--------------------Testing Set Metrics--------------------

R2-Score on Testing set ---> -3.880366289892445e+28
Residual Sum of Squares (RSS) on Training set  ---> 1.5887808172355157e+35
Mean Squared Error (MSE) on Training set       ---> 1.3272718868476202e+32
Root Mean Squared Error (RMSE) on Training set ---> 1.1520728652589876e+16

--------------------Residual Plots--------------------
```





KESIMPULAN

- Dataset kecil dengan hanya 6435 sampel & setelah prapemrosesan 7,5% sampel data dibuang.
- Memvisualisasikan distribusi data & hubungannya, membantu kami mendapatkan beberapa wawasan tentang fitur tersebut -set.
- Fitur memiliki multikolinearitas yang tinggi, oleh karena itu pada langkah Ekstraksi Fitur, kami memilih fitur yang sesuai dengan Teknik VIF.

JUMATEC

Home Product Features Contact

THANK YOU !