CONTENTS

- ABOUT
- SOURCE
- IMPORTING LIBRARIES
- IMPORTING DATA
- STATISTICAL ANALYSIS
  - DATA DESCRIPTION
    - 1.1. DATA DESCRIPTION
    - 1.2. STATISTICAL SUMMARY
  - DATA PREPROCESSING
    - 2.1. DETECTING NULL VALUES
    - 2.2. FILLING NULL VALUES
    - 2.3. DROPPING UNIQUE COLUMNS
  - CATEGORICAL COLUMNS
  - DATA VISUALIZATION
    - 3.1. BAR PLOT
    - 3.2. PIE CHART
    - 3.3. FOG ARC
    - 3.4. HEAT MAP
    - 3.5. TOP 10% AND 90 CUMSUM
    - 3.6. TOP 10% AND 90 CUMSUM
    - 3.7. TOP 10% AND SUM2 ANALYSIS
    - 3.8. TOP 10% WEIGHT AND 90 CUMSUM
    - 3.9. TOP 10% WEIGHT AND 90 CUMSUM
    - 3.10. BOX PLOT & VIOLIN PLOT
    - 3.11. BOX PLOT & VIOLIN PLOT
    - 3.12. COMPARE WITH MEAN
  - DATA PREPARATION
    - 4.1. DATA TRANSFORMATION
    - 4.2. DATA TRANSFORMATION
  - FEATURE SCALING
  - FEATURE SELECTION
  - DATA MODELING
    - 7.1. LINEAR REGRESSION
      - 7.1.1. CROSS VALIDATION
      - 7.1.2. ANOVA
      - 7.1.3. NORMALITY
      - 7.1.4. HOMOSCEDASTICITY
    - 7.2. LINEAR REGRESSION
      - 7.2.1. CROSS VALIDATION
      - 7.2.2. ANOVA
      - 7.2.3. NORMALITY
      - 7.2.4. HOMOSCEDASTICITY
    - 7.3. LINEAR REGRESSION
      - 7.3.1. CROSS VALIDATION
      - 7.3.2. ANOVA
      - 7.3.3. NORMALITY
      - 7.3.4. HOMOSCEDASTICITY
    - 7.4. PERFORMANCE OF ABOVE ON ORIGINAL DATA
    - 7.5. PERFORMANCE OF ABOVE ON TRANSFORMED DATA (BOX-COX TRANSFORMATION ANTRIC)
    - 7.6. RIDGE REGRESSION
      - 7.6.1. CROSS VALIDATION FOR TRANSFORMED DATA
      - 7.6.2. ANOVA
      - 7.6.3. NORMALITY
      - 7.6.4. HOMOSCEDASTICITY
    - 7.7. LINEAR REGRESSION
      - 7.7.1. CROSS VALIDATION FOR ORIGINAL DATA
      - 7.7.2. ANOVA
      - 7.7.3. NORMALITY
      - 7.7.4. HOMOSCEDASTICITY
    - 7.8. LINEAR REGRESSION
      - 7.8.1. CROSS VALIDATION FOR TRANSFORMED DATA
      - 7.8.2. ANOVA
      - 7.8.3. NORMALITY
      - 7.8.4. HOMOSCEDASTICITY
  - TWO STEP ANALYSIS
  - CONCLUSION
  - CONTRIBUTIONS
  - REFERENCES
  - ACKNOWLEDGMENTS

ABOUT

The Big Bear Sales Regressional Analysis Project delves into the realm of predictive analytics by forecasting the sales performance across various categories within the retail industry. By utilizing demographic and transaction data from customers, this project aims to uncover trends and patterns that influence purchasing behaviors.

Leveraging advanced statistical techniques and machine learning algorithms, the analysis will provide valuable insights into factors driving customer engagement and sales performance. The ultimate goal is to optimize marketing strategies and enhance overall business performance.

SOURCE

Website: https://paulbaumann.com/datasets/retailmarketing.html (since predictive datasets

The data scientists at Bigbear have collected 2014 sales data for 2240 products across 50 stores in 3 countries.

IMPORTING LIBRARIES

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
import scipy
import statsmodels.stats.anova
from statsmodels.graphics.mosaicplot import mosaic
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_regression import LinearRegression, Ridge, RidgeCV
from sklearn.metrics import mean_squared_error, r2_score
from statsmodels.graphics.gofplots import ProbPlot
from statsmodels.stats.diagnostic import lilliefors, jarque_bera, kstest, shapiro
from statsmodels.stats.stattools import durbinwatson
```

IMPORTING DATA

```
df = pd.read_csv('./data.csv')
Cost = df.cost_per
df.head(5)
df.shape
df.dtypes.value_counts()
```

	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	NumDealsPurchases	NumWebPurchases	NumCatalogPurchases	NumStorePurchases	NumWebVisitsMonth	AcceptedCmp3	AcceptedCmp4	AcceptedCmp5	AcceptedCmp1	AcceptedCmp2	Complain	Z_CostContact	Z_Revenue	Response
0	1957	Graduation	Single	58138.0	0	0	04-09-2012	58	635	88	546	172	88	88	3	8	10	4	7	0	0	0	0	0	0	3	11	1
1	1954	Graduation	Single	46344.0	1	1	08-03-2014	38	11	1	6	2	1	6	2	1	1	2	5	0	0	0	0	0	0	3	11	0
2	1965	Graduation	Together	71959.0	0	0	21-08-2013	26	426	49	127	111	21	42	1	8	2	10	4	0	0	0	0	0	0	3	11	0
3	1984	Graduation	Married	26646.0	1	0	10-02-2014	26	11	4	20	10	3	5	2	2	0	4	6	0	0	0	0	0	0	3	11	0
4	1981	PhD	Married	70951.0	0	0	19-01-2013	94	173	43	118	46	27	15	5	5	3	6	5	0	0	0	0	0	0	3	11	0

```
df.describe.T.round(2)
df.describe(include='object').T
```

	count	mean	std	min	25%	50%	75%	max
Year_Birth	2240.0	1968.81	11.98	1893.0	1959.0	1970.0	1977.0	1996.0
Income	2216.0	52247.25	25173.08	1730.0	35303.0	51381.5	68522.0	666666.0
Kidhome	2240.0	0.44	0.54	0.0	0.0	0.0	1.0	2.0
Teenhome	2240.0	0.51	0.54	0.0	0.0	0.0	1.0	2.0
Recency	2240.0	49.11	28.96	0.0	24.0	49.0	74.0	99.0
MntWines	2240.0	303.94	336.597392	0.0	23.75	173.5	504.25	1493.0
MntFruits	2240.0	26.302	39.773434	0.0	1.0	8.0	33.0	199.0
MntMeatProducts	2240.0	166.95	225.715373	0.0	16.0	67.0	232.0	1725.0
MntFishProducts	2240.0	37.525	54.628979	0.0	3.0	12.0	50.0	259.0
MntSweetProducts	2240.0	27.062	41.280498	0.0	1.0	8.0	33.0	263.0
MntGoldProds	2240.0	44.021	52.167439	0.0	9.0	24.0	56.0	362.0
NumDealsPurchases	2240.0	2.325	1.932238	0.0	1.0	2.0	3.0	15.0
NumWebPurchases	2240.0	4.085	2.778177	0.0	2.0	4.0	6.0	27.0
NumCatalogPurchases	2240.0	2.662	2.923116	0.0	0.0	2.0	4.0	28.0
NumStorePurchases	2240.0	5.790	3.250958	0.0	3.0	5.0	8.0	13.0
NumWebVisitsMonth	2240.0	5.316	2.454358	0.0	3.0	6.0	7.0	20.0
AcceptedCmp3	2240.0	0.073	0.260826	0.0	0.0	0.0	0.0	1.0
AcceptedCmp4	2240.0	0.074	0.262728	0.0	0.0	0.0	0.0	1.0
AcceptedCmp5	2240.0	0.073	0.260826	0.0	0.0	0.0	0.0	1.0
AcceptedCmp1	2240.0	0.064	0.245316	0.0	0.0	0.0	0.0	1.0
AcceptedCmp2	2240.0	0.013	0.114076	0.0	0.0	0.0	0.0	1.0
Complain	2240.0	0.009	0.096391	0.0	0.0	0.0	0.0	1.0
Z_CostContact	2240.0	3.000	0.000000	3.0	3.0	3.0	3.0	3.0
Z_Revenue	2240.0	11.000	0.000000	11.0	11.0	11.0	11.0	11.0
Response	2240.0	0.149	0.356274	0.0	0.0	0.0	0.0	1.0

	count	unique	top	freq
Education	2240	5	Graduation	1127
Marital_Status	2240	8	Married	864
Dt_Customer	2240	663	31-08-2012	12

```
df.shape, df.dtypes.value_counts()
```

```
(2240, 29)
int64    25
object    3
Name: count, dtype: int64
```

STATISTICAL ANALYSIS

DATA DESCRIPTION

```
Income Table 1.1
Income_Mean = df.Income.mean().round(2)
Income_Median = df.Income.median().round(2)
Income_Mode = df.Income.mode()[0].round(2)
Income_Skew = df.Income.skew().round(2)
Income_Kurt = df.Income.kurt().round(2)
Income_Range = df.Income.max() - df.Income.min()
Income_Q1 = df.Income.quantile(0.25).round(2)
Income_Q3 = df.Income.quantile(0.75).round(2)
Income_IQR = Income_Q3 - Income_Q1
Income_Outliers = df[(df.Income < (Income_Q1 - 1.5 * Income_IQR)) | (df.Income > (Income_Q3 + 1.5 * Income_IQR))].Income.count()
Income_Missing = df.Income.isna().sum()
Income_Unique = df.Income.nunique()
Income_Data_Type = df.Income.dtype
Income_Table = pd.DataFrame({'Income_Mean': [Income_Mean], 'Income_Median': [Income_Median], 'Income_Mode': [Income_Mode], 'Income_Skew': [Income_Skew], 'Income_Kurt': [Income_Kurt], 'Income_Range': [Income_Range], 'Income_Q1': [Income_Q1], 'Income_Q3': [Income_Q3], 'Income_IQR': [Income_IQR], 'Income_Outliers': [Income_Outliers], 'Income_Missing': [Income_Missing], 'Income_Unique': [Income_Unique], 'Income_Data_Type': [Income_Data_Type]})
Income_Table.T.round(2)
```

	0
Income_Mean	52247.25
Income_Median	51381.5
Income_Mode	35860.0
Income_Skew	6.76
Income_Kurt	203.03
Income_Range	664936.0
Income_Q1	35303.0
Income_Q3	68522.0
Income_IQR	33219.0
Income_Outliers	8
Income_Missing	24
Income_Unique	1730
Income_Data_Type	float64

STATISTICAL SUMMARY

```
Income Table 1.2
Income_Count = df.Income.count()
Income_Sum = df.Income.sum().round(2)
Income_Min = df.Income.min()
Income_Max = df.Income.max()
Income_Mean = df.Income.mean().round(2)
Income_Median = df.Income.median().round(2)
Income_Mode = df.Income.mode()[0].round(2)
Income_Skew = df.Income.skew().round(2)
Income_Kurt = df.Income.kurt().round(2)
Income_Range = df.Income.max() - df.Income.min()
Income_Q1 = df.Income.quantile(0.25).round(2)
Income_Q3 = df.Income.quantile(0.75).round(2)
Income_IQR = Income_Q3 - Income_Q1
Income_Outliers = df[(df.Income < (Income_Q1 - 1.5 * Income_IQR)) | (df.Income > (Income_Q3 + 1.5 * Income_IQR))].Income.count()
Income_Missing = df.Income.isna().sum()
Income_Unique = df.Income.nunique()
Income_Data_Type = df.Income.dtype
Income_Table = pd.DataFrame({'Income_Count': [Income_Count], 'Income_Sum': [Income_Sum], 'Income_Min': [Income_Min], 'Income_Max': [Income_Max], 'Income_Mean': [Income_Mean], 'Income_Median': [Income_Median], 'Income_Mode': [Income_Mode], 'Income_Skew': [Income_Skew], 'Income_Kurt': [Income_Kurt], 'Income_Range': [Income_Range], 'Income_Q1': [Income_Q1], 'Income_Q3': [Income_Q3], 'Income_IQR': [Income_IQR], 'Income_Outliers': [Income_Outliers], 'Income_Missing': [Income_Missing], 'Income_Unique': [Income_Unique], 'Income_Data_Type': [Income_Data_Type]})
Income_Table.T.round(2)
```

	0
Income_Count	2216.0
Income_Sum	115790248.0
Income_Min	1730.0
Income_Max	666666.0
Income_Mean	52247.25
Income_Median	51381.5
Income_Mode	35860.0
Income_Skew	6.76
Income_Kurt	203.03
Income_Range	664936.0
Income_Q1	35303.0
Income_Q3	68522.0
Income_IQR	33219.0
Income_Outliers	8
Income_Missing	24
Income_Unique	1730
Income_Data_Type	float64

DATA PREPROCESSING

DETECTING NULL VALUES

```
Income_Missing = df.Income.isna().sum()
Income_Missing_Percent = (Income_Missing / df.shape[0]).round(2)
Income_Missing_Table = pd.DataFrame({'Income_Missing': [Income_Missing], 'Income_Missing_Percent': [Income_Missing_Percent]})
Income_Missing_Table.T
```

	0
Income_Missing	24
Income_Missing_Percent	0.01

FILLING NULL VALUES

```
Income = df.Income.fillna(df.Income.median())
Income = df.Income.fillna(df.Income.mean())
Income_Missing = df.Income.isna().sum()
Income_Missing_Percent = (Income_Missing / df.shape[0]).round(2)
Income_Missing_Table = pd.DataFrame({'Income_Missing': [Income_Missing], 'Income_Missing_Percent': [Income_Missing_Percent]})
Income_Missing_Table.T
```

	0
Income_Missing	0
Income_Missing_Percent	0.0

DROPPING UNIQUE COLUMNS

```
Unique_Cols = df.columns[df.nunique() == df.shape[0]]
df = df.drop(Unique_Cols, axis=1)
df.shape
```

```
(2240, 28)
```

```
Const_Cols = df.columns[df.nunique() == 1]
df = df.drop(Const_Cols, axis=1)
df.shape
```

```
(2240, 27)
```

CATEGORICAL COLUMNS

```
Cat = df.select_dtypes('object')
Cat.shape, Cat.dtypes.value_counts()
```

```
(2240, 3)
object    3
Name: count, dtype: int64, Name: count, dtype: int64
```

DATA VISUALIZATION

BAR PLOT

```
plt.clear
plt.figure(figsize=(8, 6))
sns.countplot('Education')
plt.show()
```

```
For Non-Premium
```

Product Count
Product Count
Product Count

```
The application of the column-from_fixedfactory_service variable transmission is :-
Mean : 0.000000
Std : 0.000000
Max : 0.000000
Min : 0.000000
Skew : 0.000000
Kurt : 0.000000
The application of the column-from_fixed_fact
```

	Sex_Weight	Sex_Fat_Content	Sex_Fiber	Sex_Sugar	Sex_BMI	Sex_Sodium_Intake	Sex_Cholesterol_Intake	Sex_Protein_Intake	Sex_Water_Intake
	0.125000	Regular	0.333333	Low	0.125000	0.667	0.677	0.667	Medium
	0.125000	Low Fat	0.333333	Medium	0.125000	0.667	0.677	0.667	Medium
	0.125000	Low Fat	0.333333	Medium	0.125000	0.667	0.677	0.667	Medium
	0.125000	Regular	0.333333	Medium	0.125000	0.667	0.677	0.667	Medium
	0.125000	Regular	0.333333	Medium	0.125000	0.667	0.677	0.667	Medium
	0.125000	Low Fat	0.333333	Medium	0.125000	0.667	0.677	0.667	Medium
50%	0.125000	Low Fat	0.333333	Medium	0.125000	0.667	0.677	0.667	Medium
75%	0.125000	Low Fat	0.333333	Medium	0.125000	0.667	0.677	0.667	Medium
max	0.125000	Low Fat	0.333333	Medium	0.125000	0.667	0.677	0.667	Medium
count	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
mode	0.125000	Low Fat	0.333333	Medium	0.125000	0.667	0.677	0.667	Medium
freq	0.125000	Low Fat	0.333333	Medium	0.125000	0.667	0.677	0.667	Medium

5000 rows × 10 columns

CHECKING FOR ASSUMPTIONS:

LINEARITY

```
from sklearn.linear_sklearn import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
from scipy.stats import shapiro, kstest, probplot
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Prepare data
X = df[['Sex_Weight', 'Sex_Fat_Content', 'Sex_Fiber', 'Sex_Sugar', 'Sex_BMI', 'Sex_Sodium_Intake', 'Sex_Cholesterol_Intake', 'Sex_Protein_Intake', 'Sex_Water_Intake']]
y = df['Sex_Calories']

# Create polynomial features
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Fit linear regression
model = LinearRegression()
model.fit(X_poly, y)

# Predictions
y_pred = model.predict(X_poly)

# Residuals
residuals = y - y_pred

# Plot residuals vs predicted values
plt.figure(figure=(10, 6))
sns.scatterplot(x=y_pred, y=residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Predicted Values')
plt.show()
```

OUTPUT:

ADABOOST REGRESSOR

```
ada_model=AdaBoostRegressor(n_estimators=50 ,
learning_rate=1.0 ,random_state=1 ) ,"model", ( ) ,"score", 10 ) ]
[AdaBoostRegressor(base_estimator=None,
learning_rate=1.0,loss='linear',n_estimators=50,
random_state=1), 0.7126171146498428]
for ,m,s in ada_model:
print ("The mean value of the Training Data is:", np.mean(y_train),sep=" ")
MY_pred=m.predict(X_train) ,"Training", ( ) ,"Mean", ( ) ,"Root", ( )
print ("The mean value of the Testing Data is:", np.mean(y_test),sep=" ")
MY_pred=m.predict(X_test) ,"Testing", ( ) ,"Mean", ( ) ,"Root", ( )
The mean value of the Training Data is: 0.5
The mean value of the Testing Data is: 0.5
MY_pred=m.predict(X_train) ,"Training", ( ) ,"Mean", ( )
MY_pred=m.predict(X_test) ,"Testing", ( ) ,"Mean", ( )
print ("The Root Mean Squared Error of Training Data is:", ( ) ,"Mean", ( ) ,sqrt(mean_squared_error(y_train,MY_pred)),sep=" ")
print ("The Root Mean Squared Error of Testing Data is:", ( ) ,"Mean", ( ) ,sqrt(mean_squared_error(y_test,MY_pred)),sep=" ")
print ("Coefficient Determination of Training Data is:", ( ) ,"Mean", ( ) ,m.score(X_train,y_train),sep=" ")
print ("Coefficient Determination of Testing Data is:", ( ) ,"Mean", ( ) ,m.score(X_test,y_test),sep=" ")
The mean value of the Training Data is: 0.5
The mean value of the Testing Data is: 0.5
The Root Mean Squared Error of Training Data is: 0.0
The Root Mean Squared Error of Testing Data is: 0.0
Coefficient Determination of Training Data is: 1.0
Coefficient Determination of Testing Data is: 1.0
ada_model=AdaBoostRegressor(base_estimator=None,loss='linear',
learning_rate=1.0,n_estimators=50,random_state=1)
for ,m,s in ada_model:
print ("Cross Validation Score of Training Data:", ( ) ,"Mean", ( ) ,cross_val_score(m,X_train,y_train,cv=10),sep=" ")
print ("Cross Validation Score of Testing Data:", ( ) ,"Mean", ( ) ,cross_val_score(m,X_test,y_test,cv=10),sep=" ")
```

Cross Validation Score of Training Data:	Mean	Cross Validation Score of Testing Data:	Mean
1.0000000	1.0000000	1.0000000	1.0000000