PROJECT ANALYSIS

Topic

“Predicting Walmart Sales”

Centre for Intelligent Cyber Physical Systems

Indian Institute of Technology

Guwahati

Submitted to:

Prof. Rhythm Grover

Submitted by:

PRABHAT KUMAR

Roll-no. 214156006

MTech Robotics & AI

Contents

1. Statement of Research:

For a big sales company it becomes very difficult to fulfil demand of customers when the demand beforehand is unknown. As a result, the aim is to predict the weekly sales of WALMART. Since sales depend on several factors, the strong aim is to determine the way these factors influence the sales of products. This is done to study the market potential and customer power to purchase the products. Thus, this will help in maintaining proper inventory with good supplies to all the warehouses.

2. Description of Dataset

This is the historical data that covers sales from 2010-02-05 to 2012-11-01, in the file Walmart$Store$sales. Within this file you will find the following fields:

- Store - the store number
- Date - the week of sales
- Weekly_Sales - sales for the given store
- Holiday_Flag - whether the week is a special holiday week 1 – Holiday week 0 – Non-holiday week
- Temperature - Temperature on the day of sale
- Fuel_Price - Cost of fuel in the region
- CPI – Prevailing consumer price index
- Unemployment - Prevailing unemployment rate
- Holiday Events\
  Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13\
  Labour Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13\
  Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13\
  Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13

The dataset consists of:

- 6435 rows and 8 columns
- There are distinct 45 Walmart stores located in different regions are available.
- There is holiday flag column that takes Boolean yes or no.
- This takes Unemployment rate as a feature.
- Temperature for the day is also an input feature.
- Fuel price is another important feature considered.
- Consumer price index is also one feature.
- We predict weekly sales from the provided above features. We see if all or some input variable columns are essential for predicting weekly sales **OR** taking unnecessary column is adding variability in the predicted output.

2.1 Head of the dataset

	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

There are in total 45 stores numbered 1 to 45.

2.2 Quick Statistic Summary about this data

```
df.describe()
```

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000
mean	23.000000	1.046965e+06	0.069930	60.663782	3.358607	171.578394	7.999151
std	12.988182	5.643666e+05	0.255049	18.444933	0.459020	39.356712	1.875885
min	1.000000	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	3.879000
25%	12.000000	5.533501e+05	0.000000	47.460000	2.933000	131.735000	6.891000
50%	23.000000	9.607460e+05	0.000000	62.670000	3.445000	182.616521	7.874000
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000	212.743293	8.622000
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000	227.232807	14.313000

The count of unique values in each column is shown below:

```
Store: 45
Fuel_Price: 892
Unemployment: 349
Day: 7
Month: 12
Year: 3
```

The following dataset consists of in total 8 columns and 6435 rows.

The target column is weekly sales

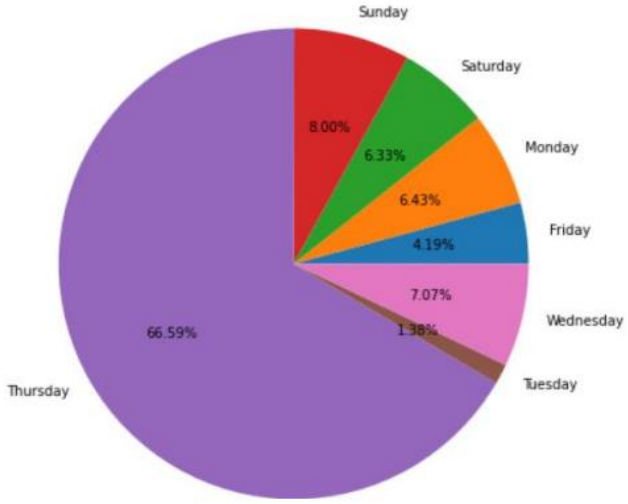
#	Column	Non-Null Count	Dtype
0	Store	6435 non-null	int64
1	Date	6435 non-null	object
2	Weekly_Sales	6435 non-null	float64
3	Holiday_Flag	6435 non-null	int64
4	Temperature	6435 non-null	float64
5	Fuel_Price	6435 non-null	float64
6	CPI	6435 non-null	float64
7	Unemployment	6435 non-null	float64

3. Exploratory Data Analysis:

We use exploratory data analysis to study and investigate data sets and describe their primary properties, frequently using visualization techniques. It assists to determine how to best manipulate data sources to obtain the answers, making it easier to find patterns, test hypothesis, and verify assumptions.

This is primarily used to examine what data can disclose outside of formal modelling or hypothesis testing tasks, and to gain a better knowledge of dataset variables and their interactions. It might also assist you in determining whether the statistical techniques you are contemplating for data analysis are suitable.

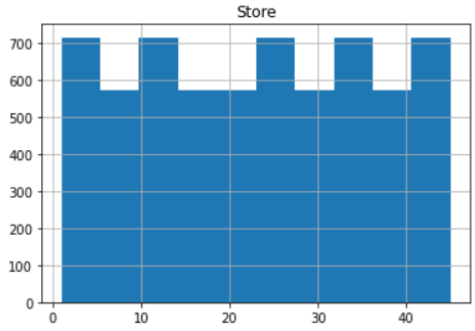
3.1 Pie plot to show proportion of sales on specific day of the week:



3.2 Histogram plot of input variables

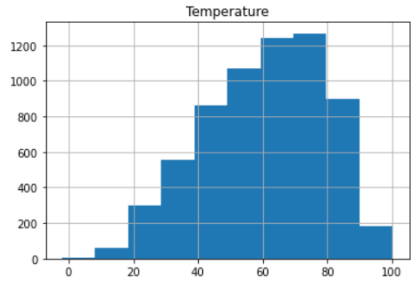
```
import numpy as np
df.hist(column = 'Store',xlabelsize =10)
```

```
array([[<AxesSubplot:title={'center':'Store'}>]],
```



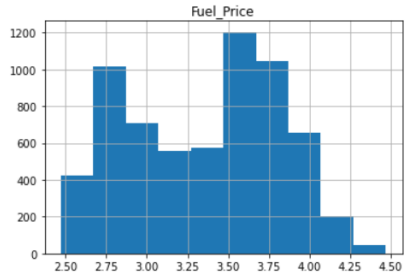

```
df.hist(column = 'Temperature',xlabelsize =10)
```

```
array([[<AxesSubplot:title={'center':'Temperature'}>
```



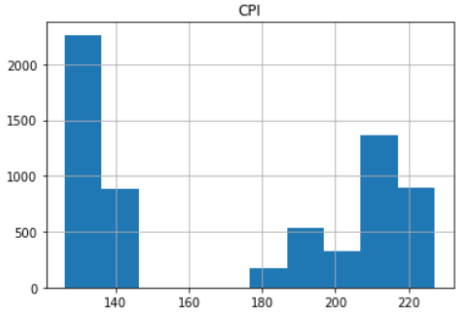
```
df.hist(column = 'Fuel_Price',xlabelsize =10)
```

```
array([[<AxesSubplot:title={'center':'Fuel_Price'}
```



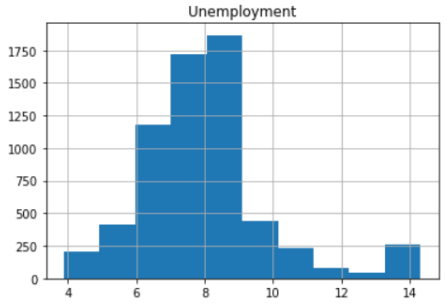

```
df.hist(column = 'CPI',xlabelsize =10)
```

```
array([[<AxesSubplot:title={'center':'CPI'}>]], dty
```



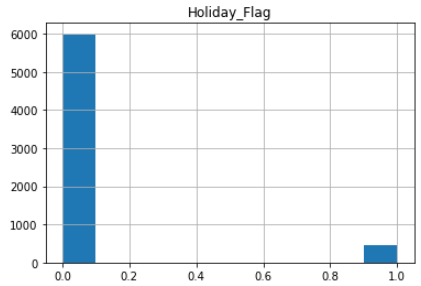

```
df.hist(column = 'Unemployment',xlabelsize =10)
```

```
array([[<AxesSubplot:title={'center':'Unemployment
```




```
df.hist(column = 'Holiday_Flag',xlabelsize =10)
```

```
array([[<AxesSubplot:title={'center':'Holiday_Flag'
```



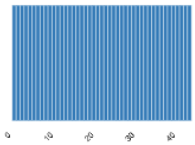
Variables :

Store

Real number (ℝ≥0)

HIGH CORRELATION

Distinct	45	Minimum	1
Distinct (%)	0.7%	Maximum	45
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	23	Memory size	50.4 KiB



Date

Categorical

HIGH CARDINALITY
UNIFORM

Distinct	143
Distinct (%)	2.2%
Missing	0
Missing (%)	0.0%
Memory size	50.4 KiB

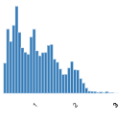
01-10-2010	45
20-01-2012	45
14-01-2011	45
18-02-2011	45
03-06-2011	45
Other values (138)	6210

Weekly_Sales

Real number (ℝ≥0)

HIGH CORRELATION
UNIQUE

Distinct	6435	Minimum	209986.25
Distinct (%)	100.0%	Maximum	3818686.45
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	1046964.878	Memory size	50.4 KiB



Holiday_Flag

Categorical

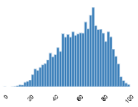
Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	50.4 KiB

0	5985
1	450

Temperature

Real number (ℝ)

Distinct	3528	Minimum	-2.06
Distinct (%)	54.8%	Maximum	100.14
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	1
Infinite (%)	0.0%	Negative (%)	< 0.1%
Mean	60.66378244	Memory size	50.4 KiB



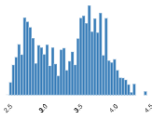
Fuel_Price

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION Fuel_Price

Distinct	892
Distinct (%)	13.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	3.358606838

Minimum	2.472
Maximum	4.468
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	50.4 KiB



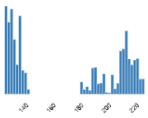
CPI

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION

Distinct	2145
Distinct (%)	33.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	171.5783938

Minimum	126.064
Maximum	227.2328068
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	50.4 KiB



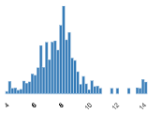
Unemployment

Real number ($\mathbb{R}_{\geq 0}$)

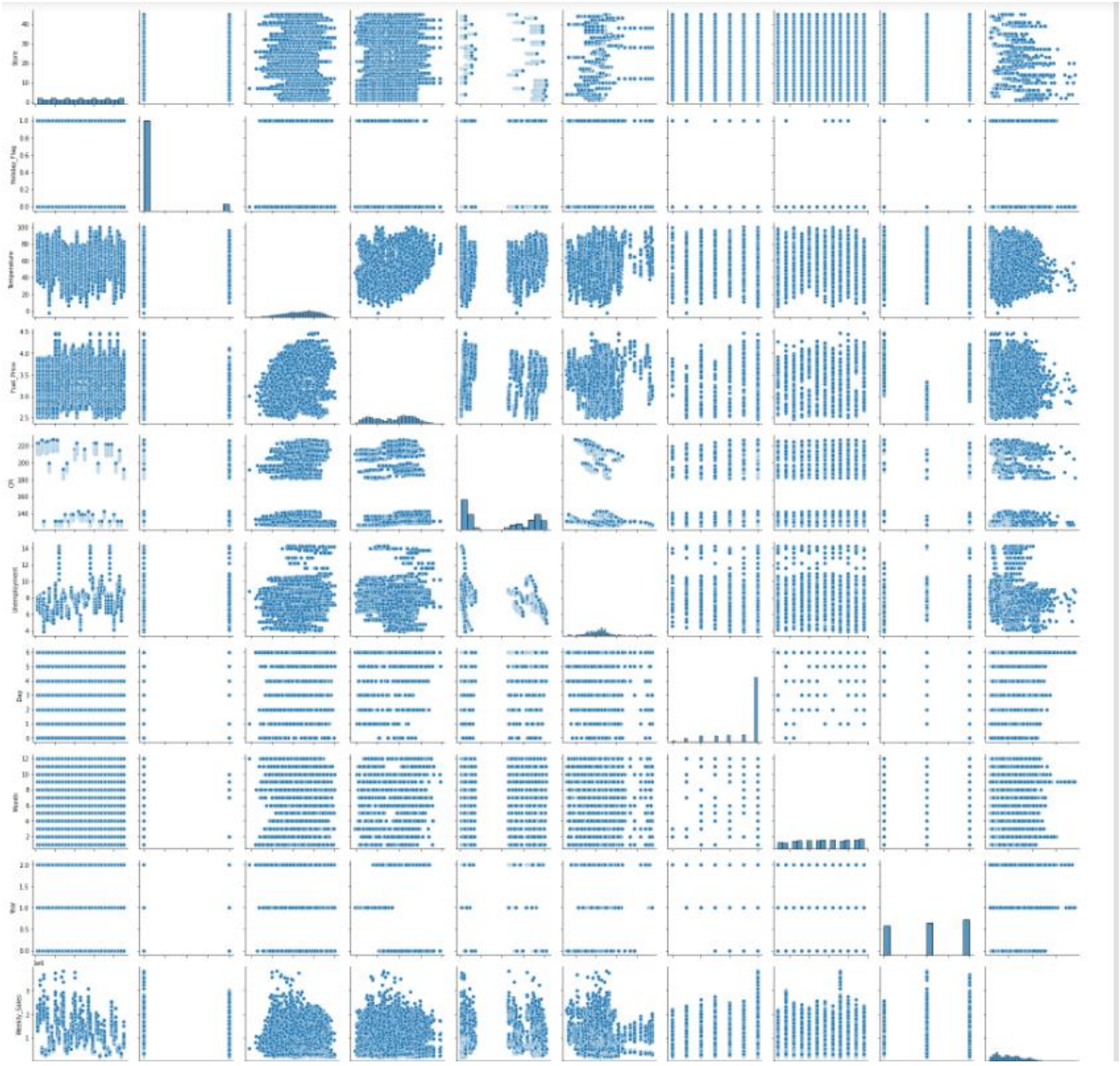
HIGH CORRELATION

Distinct	349
Distinct (%)	5.4%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	7.999151049

Minimum	3.879
Maximum	14.313
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	50.4 KiB



3.3 Pair plot for columns



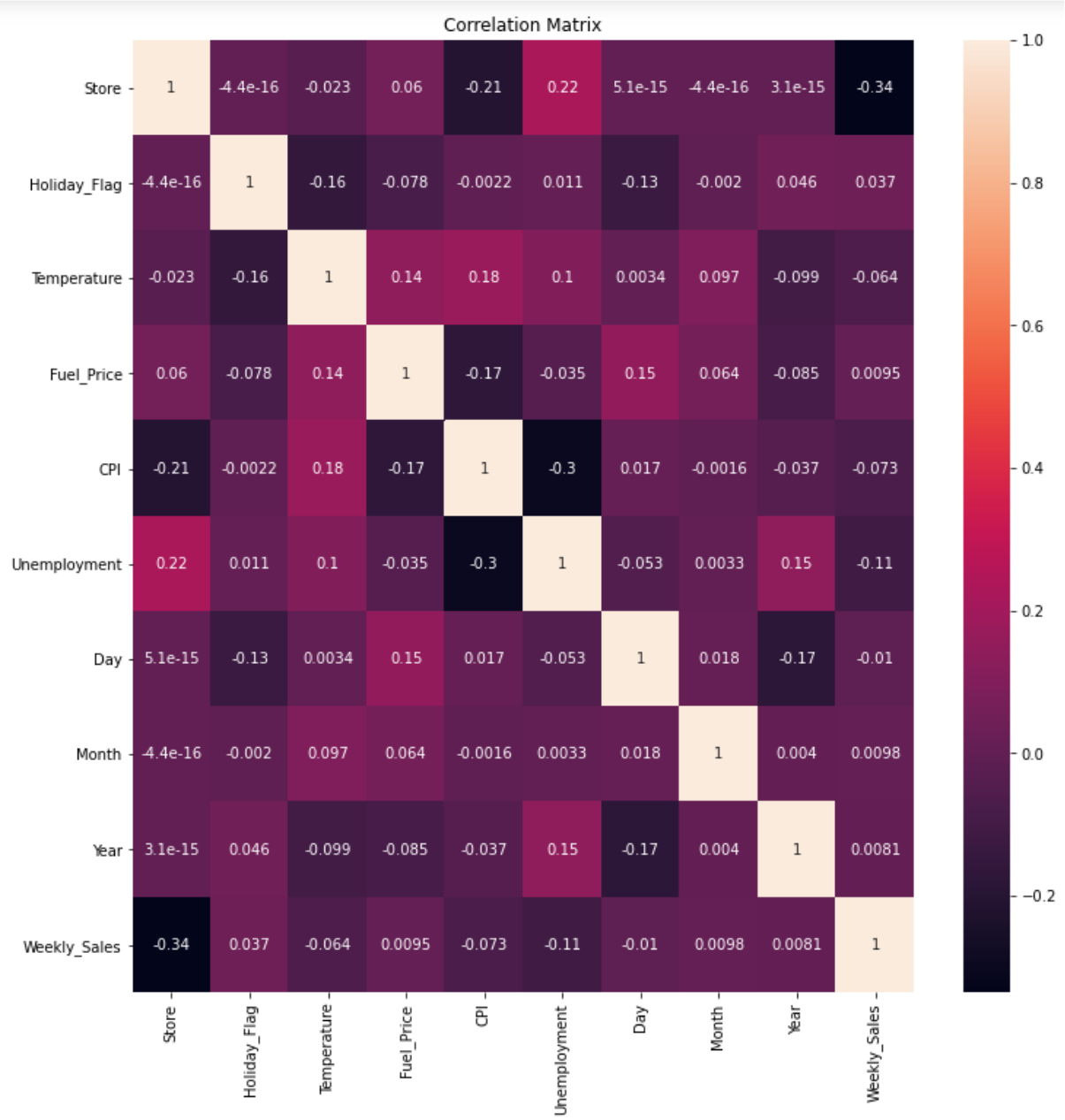
A pair plot is a pairwise relationships in a dataset. The pair plot function creates a grid of axes such that each variable in data variable in data will be shared in the y-axis across a single row and in the x-axis across a single column.

From pair plot we can see that in what manner the features vary with each other.

3.4 Heat Map of Sum Important columns

A heatmap contains values representing various shades of the same colour for each value to be plotted. Usually, the darker shades of the chart represent higher values than the lighter shade. For a very different value a completely colour can also be used.

Heat Map can also be used for finding that which parameters are highly correlated and which are loosely correlated so that we can reduce our model into reduced fit model.



3.5 Cleaning of data

After we have the data, we need to clean and prepare the data according to the model's requirements. In any machine learning problem, this is the step that is the most important and the most time consuming. We used various techniques and logistics.

Handling of categorical features

Categorical data are variables that contain label values rather than numeric values. Many algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric.

This means that categorical data must be converted to a numerical form. If the categorical variable is an output variable, you may also want to convert predictions by the model back to categorical form in order to present them or use them in some application.

Categorical data can be converted to numerical data involving two steps:

1. Integer Encoding

2. One-Hot Encoding

1.) Integer Encoding

At first step, each unique category value is assigned an integer value. This is called a label encoding and is easily reversible. The integer values have a natural ordered relationship between each other and algorithms, may be able to understand and harness this relationship.

Day	Month	Year
Saturday	May	2010
Wednesday	December	2010
Thursday	February	2010
Thursday	February	2010
Sunday	May	2010

The days, month and year are categorical variables and each unique values in these variables are assigned integers.

```
tge= {'Sunday':5,'Monday':3, 'Tuesday':0, 'Wednesday':4, 'Thursday':6,  'Friday':1,'Saturday':2}
df['Day'] = df['Day'].map(tge)
df.head()
```

```
yr = {2010:1, 2011:2, 2012:0}
df['Year'] = df['Year'].map(yr)
df.head()
```

```
mn = {'January':1,'February':3,'March':4,'April':12,'May':11,'June':6,'July':8,'August':5,'September':7,'October':10,'Novemenber
df['Month'] = df['Month'].map(mn)
df.head()
```

Finally, our categorical variables column get transformed to integer coded labels.

Day	Month	Year
2	11	1
4	9	1
6	3	1
6	3	1
5	11	1

2.) One Hot Encoding

For categorical variables where no such ordinal relationship exists, the integer encoding is not enough. In fact, using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories).

In this case, a one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
transformer = ColumnTransformer(transformers=[('tf',OneHotEncoder(sparse=False,drop='first'),['Store'])],remainder='passthrough')
```

Here store column is also a label data hence we applied One Hot encoder to store variable and the Date column is split into Day, month, year and is encoded with integers.

4. MODEL SELECTION

$$y_i = \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i, \qquad \mathbb{E}[\epsilon_i] = \sigma^2$$

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}, \quad \mathbb{E}[\hat{\boldsymbol{\beta}}|\mathbf{X}] = \boldsymbol{\beta}, \quad Cov\,[\hat{\boldsymbol{\beta}}|\boldsymbol{X}] = \sigma^2(\boldsymbol{X}^\intercal\boldsymbol{X})^{-1}$$

If $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, model correct and $\mathbf{X}$ fixed, $\hat{\boldsymbol{\beta}} \sim \mathcal{N}(\boldsymbol{\beta}, \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1})$

test the null hypothesis that $\beta_j = 0$.

$$\hat{\sigma}^2 = \frac{\|\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\|^2}{N-p}, \quad (N-p)\frac{\hat{\sigma}^2}{\sigma^2} \sim \chi^2_{N-p}$$

Σ_j^2 =j-th diagonal element of $(\boldsymbol{X}^\intercal\boldsymbol{X})^{-1}$

$$T_j = \frac{\frac{\hat{\beta}_j - 0}{\sigma\Sigma_j}}{\sqrt{\frac{(N-p)\frac{\hat{\sigma}^2}{\sigma^2}}{N-p}}} = \frac{\hat{\beta}_j}{\hat{\sigma}\Sigma_j}$$ is t distributed with $N-p$ degrees of freedom

can be used as a t-test to test the hypothesis that $\hat{\beta}_j = 0$

$$\hat{\beta} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

$\hat{\beta}$ = ordinary least squares estimator
$\boldsymbol{X}$ = matrix regressor variable X
$\top$ = matrix transpose
$\boldsymbol{y}$ = vector of the value of the response variable

The dataset is of linear regression and we choose the model to be Linear regression.

Regression analysis computed using a variety of algorithms and then predicts the continuous value. The input consists of a series of variables, with the continuous range value serving as the goal variable.

• Linear regression: To predict the continuous values, Linear regression is used. Certain known parameters are given to the machine learning algorithms, it predicts the continuous values as output. It cannot used for the classification problems. The proposed model predicts the score using the Linear Regression

• Ridge regression: Ridge regression is also used to predict the continuous values. When the variables used for the prediction greater than the observations of when multicollinearity present in the data, ridge regression is used set has multicollinearity (correlations between predictor variables).
• Lasso regression: Lasso regression is a type of linear regression that used for predicting the continuous values. Shrinkage is used in the lasso regression. When data values focus towards central point shrinkage occurs. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models.

5. Lasso model selection: AIC-BIC / cross validation

This example focuses on model selection for Lasso models that are linear models with an L1 penalty for regression problems.

Selecting Lasso via an information criterion

`LassoLarsIC` provides a Lasso estimator that uses the Akaike information criterion (AIC) or the Bayes information criterion (BIC) to select the optimal value of the regularization parameter alpha.

Before fitting the model, we will standardize the data with a `StandardScaler`. In addition, we will measure the time to fit and tune the hyperparameter alpha in order to compare with the cross-validation strategy.

We will first fit a Lasso model with the AIC criterion.

```
import time
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LassoLarsIC
from sklearn.pipeline import make_pipeline

start_time = time.time()
lasso_lars_ic = make_pipeline(
    StandardScaler(), LassoLarsIC(criterion="aic", normalize=False)
).fit(X_train, Y_train)
fit_time = time.time() - start_time
```

We store the AIC metric for each value of alpha used during `fit`.

```
results = pd.DataFrame(
    {
        "alphas": lasso_lars_ic[-1].alphas_,
        "AIC criterion": lasso_lars_ic[-1].criterion_,
    }
).set_index("alphas")
alpha_aic = lasso_lars_ic[-1].alpha_
```

Now, we perform the same analysis using the BIC criterion.

```
lasso_lars_ic.set_params(lassolarsic__criterion="bic").fit(X_train, Y_train)
results["BIC criterion"] = lasso_lars_ic[-1].criterion_
alpha_bic = lasso_lars_ic[-1].alpha_
```

```
def highlight_min(x):
    x_min = x.min()
    return ["font-weight: bold" if v == x_min else "" for v in x]


results.style.apply(highlight_min)
```

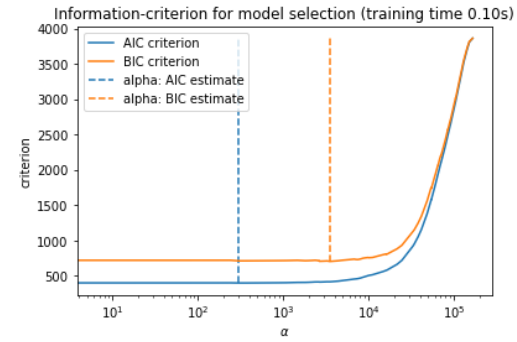
alphas	AIC criterion	BIC criterion
166817.5578375079	3861.000000	3861.000000
153794.00834522396	3812.436146	3818.694828
152629.4654786539	3805.597585	3818.114948
144394.02252085585	3714.480588	3733.256633
133268.1718978638	3555.844052	3580.878778
130066.53438489449	3501.764296	3533.057703
122207.5202562995	3341.667623	3379.219712
105239.78160481909	2965.499609	3009.310379
105112.5958315477	2964.661521	3014.730973
102633.24029315518	2907.518107	2963.846241
92139.63074749496	2656.402497	2718.989312

59918.39588123962	1758.192548	1908.400904
55182.38318151111	1598.311563	1754.778600
54414.85185515428	1574.141207	1736.866925
54395.665951627896	1575.458797	1744.443197
54123.45617820137	1567.553600	1742.796682
48808.18163466272	1380.908352	1562.410116
41787.16628620607	1156.860207	1344.620652
38234.77070003835	1054.044280	1248.063407
34450.11772561076	951.764260	1152.042068
33255.52201636207	922.013029	1128.549519
24651.797007428057	720.414804	933.209975
21626.78516643246	664.098342	883.152194
16280.475170860303	579.299547	804.612081

4360.619339222223	426.321836	714.221185
3624.227336898728	418.711077	706.610426
3524.7197698734262	417.948669	**705.848018**
3297.8605471462492	418.274947	712.432978
2965.708691861496	415.839795	709.997825
2665.0715470228843	413.864192	708.022222
2651.4719594443723	415.767058	716.183770
2378.0671466061817	415.456342	722.131735
2199.2474524949175	412.997590	719.672984
2133.5610939217227	412.181496	718.856889
1964.0925498721715	410.164280	716.839673
1817.7120595560793	409.004400	715.679794
1469.4324222440669	408.489840	721.423914

841.6439980819608	403.712982	716.647057
368.8241610074716	401.160261	714.094336
298.4780483937356	**401.037530**	713.971605
243.30885219415507	402.875147	722.067903
174.1706132260618	402.642659	721.835416
83.07853650082933	402.507445	721.700201
80.5523522391894	402.500909	721.693666
54.43431183705221	402.479718	721.672474
18.179401940360524	402.407398	721.600154
13.758800970106822	402.406535	721.599291
8.28978874181812	402.401301	721.594058
6.393129624920045	402.401106	721.593863
0.0	404.398319	729.849756

```
ax = results.plot()
ax.vlines(
    alpha_aic,
    results["AIC criterion"].min(),
    results["AIC criterion"].max(),
    label="alpha: AIC estimate",
    linestyles="--",
    color="tab:blue",
)
ax.vlines(
    alpha_bic,
    results["BIC criterion"].min(),
    results["BIC criterion"].max(),
    label="alpha: BIC estimate",
    linestyle="--",
    color="tab:orange",
)
ax.set_xlabel(r"$\alpha$")
ax.set_ylabel("criterion")
ax.set_xscale("log")
ax.legend()
_ = ax.set_title(
    f"Information-criterion for model selection (training time {fit_time:.2f}s)
)
```



After splitting training and testing dataset we fit the model using training data and tested using test data and we predict the R squared score.

I used One Hot Encoding for the STORE variable which is required otherwise the R2 score would come less. This is the method of data pre processing which is very much required in order to perform best linear regression results.

```
def pred_model(model,X_train,Y_train,X_test,Y_test):
    c = model()
    c.fit(X_train,Y_train)
    y_pred = c.predict(X_test)
    print(model)
    print(f'MSE: {mean_squared_error(Y_test,y_pred)}')
    print(f'MAE: {mean_absolute_error(Y_test,y_pred)}')
    print(f'R2 : {r2_score(Y_test,y_pred)}')



pred_model(LinearRegression,X_train,Y_train,X_test,Y_test)
```

```
<class 'sklearn.linear_model._base.LinearRegression'>
MSE: 25913530442.619396
MAE: 94445.49920330281
R2 : 0.9184356055757809
```

After fitting Linear regression model, we obtain the Ordinary Least Squares regression results. Through this we obtain the coefficients of each variable column and then we do null hypothesis testing. The variables with p-value greater than 0.05(significance level) we reject those variables and then again fit the model without these variables as input variables.

Here using stats model STORE variable is not one hot encoded and give results of R2 as 0.14.

```
X2 = sm.add_constant(X)
est = sm.OLS(Y, X2)
est2 = est.fit()
print(est2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Weekly_Sales   R-squared:                       0.142
Model:                            OLS   Adj. R-squared:                  0.141
Method:                 Least Squares   F-statistic:                     118.0
Date:                Sat, 23 Apr 2022   Prob (F-statistic):          6.70e-206
Time:                        21:13:21   Log-Likelihood:                -93860.
No. Observations:                6435   AIC:                         1.877e+05
Df Residuals:                    6425   BIC:                         1.878e+05
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const         1.989e+06   7.81e+04     25.468      0.000    1.84e+06    2.14e+06
Store        -1.537e+04    522.211    -29.441      0.000   -1.64e+04   -1.44e+04
Holiday_Flag  7.011e+04   2.61e+04      2.682      0.007    1.89e+04    1.21e+05
Temperature   -995.9156    380.265     -2.619      0.009   -1741.361    -250.470
Fuel_Price    1.084e+04    1.5e+04      0.722      0.470   -1.86e+04    4.02e+04
CPI          -2314.3579    184.977    -12.512      0.000   -2676.975   -1951.741
Unemployment -2.232e+04   3835.956     -5.818      0.000   -2.98e+04   -1.48e+04
Day          -2693.0412   4202.221     -0.641      0.522   -1.09e+04    5544.712
Month         2077.5941   1928.568      1.077      0.281   -1703.042    5858.230
Year          5417.0836   8304.597      0.652      0.514   -1.09e+04    2.17e+04
==============================================================================
Omnibus:                      187.390   Durbin-Watson:                   0.129
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              203.409
Skew:                           0.433   Prob(JB):                     6.77e-45
Kurtosis:                       3.096   Cond. No.                     2.27e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.27e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Now here since, columns ‘Fuel_Price’, ‘Day’, ‘Month’ and ‘Year’ have p-values greater than the significance level. Hence, we reject these columns and fit the linear regression line again.

Now since the ‘Day’ ‘Month’ and ‘Year’ variables came originally from ‘Date’ variable. Hence we reject ‘Date’ and ‘Fuel_Price’ from the original dataset and fit linear regression model again.

```
df.pop('Date')
```

```
0        05-02-2010
1        12-02-2010
2        19-02-2010
3        26-02-2010
4        05-03-2010
            ...
6430     28-09-2012
6431     05-10-2012
6432     12-10-2012
6433     19-10-2012
6434     26-10-2012
Name: Date, Length: 6435, dtype: object
```

```
df.pop('Fuel_Price')
```

```
0        2.572
1        2.548
2        2.514
3        2.561
4        2.625
         ...
6430     3.997
6431     3.985
6432     4.000
6433     3.969
6434     3.882
Name: Fuel_Price, Length: 6435, dtype: float64
```

6. The new dataset after removing the uncorrelated features

	Store	Weekly_Sales	Holiday_Flag	Temperature	CPI	Unemployment
0	1	1643690.90	0	42.31	211.096358	8.106
1	1	1641957.44	1	38.51	211.242170	8.106
2	1	1611968.17	0	39.93	211.289143	8.106
3	1	1409727.59	0	46.63	211.319643	8.106
4	1	1554806.68	0	46.50	211.350143	8.106
...	...	...	...	...	...	...
6430	45	713173.95	0	64.88	192.013558	8.684
6431	45	733455.07	0	64.89	192.170412	8.667
6432	45	734464.36	0	54.47	192.327265	8.667
6433	45	718125.53	0	56.47	192.330854	8.667
6434	45	760281.43	0	58.85	192.308899	8.667

6435 rows × 6 columns

```
X = df.iloc[:,:-1]
X
```

	Store	Holiday_Flag	Temperature	CPI	Unemployment
0	1	0	42.31	211.096358	8.106
1	1	1	38.51	211.242170	8.106
2	1	0	39.93	211.289143	8.106
3	1	0	46.63	211.319643	8.106
4	1	0	46.50	211.350143	8.106
...	...	...	...	...	...
6430	45	0	64.88	192.013558	8.684
6431	45	0	64.89	192.170412	8.667
6432	45	0	54.47	192.327265	8.667
6433	45	0	56.47	192.330854	8.667
6434	45	0	58.85	192.308899	8.667

6435 rows × 5 columns

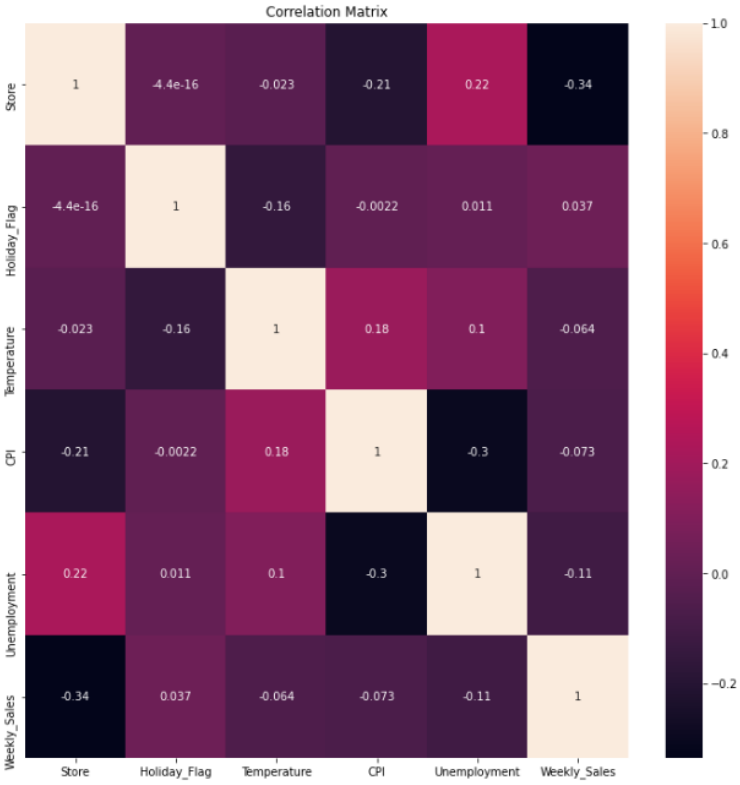
```
Y = df.iloc[:,-1]
Y
```

```
0       1643690.90
1       1641957.44
2       1611968.17
3       1409727.59
4       1554806.68
           ...
6430     713173.95
```

lick to scroll output; double click to hide

```
6433     718125.53
6434     760281.43
Name: Weekly_Sales, Length: 6435, dtype: float64
```

7. Heat Map after conducting null Hypothesis



From this correlation matrix we can see that our all-input variables are well correlated to the output variable I.e., weekly sales.

8. New Model fitting

Now since we retained the 'Store' column which contains label data. We do One Hot Encoding on Store variable and thus this increases the number of column variables.

Thus, our regression coefficients obtained are for all the variables including the One Hot Encoded variable.

Then, we again check for the null hypothesis to see the variables after fitting lie within the confidence interval.

```python
def pred_model(model,X_train,Y_train,X_test,Y_test):
    c = model()
    c.fit(X_train,Y_train)
    y_pred = c.predict(X_test)
    print(model)
    print(f'MSE: {mean_squared_error(Y_test,y_pred)}')
    print(f'MAE: {mean_absolute_error(Y_test,y_pred)}')
    print(f'R2 : {r2_score(Y_test,y_pred)}')
    print("coefficients are",c.coef_)



pred_model(LinearRegression,X_train,Y_train,X_test,Y_test)
```

```
<class 'sklearn.linear_model._base.LinearRegression'>
MSE: 26330653602.6347
MAE: 93099.20247109151
R2 : 0.917122685361289
coefficients are [ 3.83824279e+05 -1.15883431e+06  4.80433314e+05 -1.26822958e+
06
 -2.07252426e+04 -1.00507988e+06 -6.89368541e+05 -1.04396514e+06
  3.31797043e+05 -2.14213325e+05 -5.05269430e+05  4.00584157e+05
  4.21672232e+05 -9.72878961e+05 -1.09955967e+06 -7.26264108e+05
 -5.06299602e+05 -1.44474697e+05  5.36988780e+05 -8.14162236e+05
 -5.86541494e+05 -2.51937292e+05 -2.34214840e+05 -8.71945020e+05
 -6.05549807e+05  2.10865466e+05 -1.95369045e+05 -1.04072973e+06
 -1.12818023e+06 -1.62458708e+05 -4.10086391e+05 -1.32069495e+06
 -6.07296249e+05 -6.69632936e+05 -1.18793240e+06 -1.03844379e+06
 -1.13004696e+06 -1.10804610e+05 -6.84350377e+05 -3.57043004e+05
 -1.02595535e+06 -8.98563772e+05 -1.32023770e+06 -7.89133825e+05
  7.69299359e+04 -8.45504565e+02 -3.50569419e+02 -1.51154939e+04]
```

9. The regression score

The regression score we see is the R squared score that comes out to be **91.71%.**

10. Ordinary Least Squares Regression Results after performing Null Hypothesis:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Weekly_Sales   R-squared:                       0.141
Model:                            OLS   Adj. R-squared:                  0.141
Method:                 Least Squares   F-statistic:                     211.9
Date:                Fri, 29 Apr 2022   Prob (F-statistic):          7.51e-210
Time:                        12:00:22   Log-Likelihood:                -93861.
No. Observations:                6435   AIC:                         1.877e+05
Df Residuals:                    6429   BIC:                         1.878e+05
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         2.032e+06   5.07e+04     40.114      0.000    1.93e+06    2.13e+06
Store        -1.537e+04    521.337    -29.488      0.000   -1.64e+04   -1.44e+04
Holiday_Flag  7.222e+04   2.59e+04      2.787      0.005    2.14e+04    1.23e+05
Temperature   -929.0252    369.081     -2.517      0.012   -1652.547    -205.503
CPI          -2345.9264    180.191    -13.019      0.000   -2699.160   -1992.693
Unemployment  -2.22e+04   3755.948     -5.910      0.000   -2.96e+04   -1.48e+04
==============================================================================
Omnibus:                      188.685   Durbin-Watson:                   0.130
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              204.924
Skew:                           0.434   Prob(JB):                     3.17e-45
Kurtosis:                       3.100   Cond. No.                     1.46e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.46e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Here all the column variables with p-values are less than significance level therefore

We get the line of best fit with the variables which are best correlated.

11. Residual plot

Residual plot is the plot between the residue i.e., **actual weekly sales–predicted weekly sales** and the actual weekly sales.

```
residue = Y_test - y_pred
residue
```

```
2436     57167.870467
3361    -40213.038744
233    -179237.895507
3667     70699.413533
5011     60988.053758
            ...
4787       520.075762
869    -160863.397675
2358     -7843.123555
2006    -31585.587530
6266     14166.139057
Name: Weekly_Sales, Length: 2574, dtype: float64
```

```
import matplotlib.pyplot as plt
plt.scatter(Y_test,residue)
plt.title('residual plot')
plt.xlabel("Weekly_Sales")
plt.ylabel("Residual Error")
```

```
Text(0, 0.5, 'Residual Error')
```