BUSINESS ANALYTICS 4TH ASSIGNMENT

16102275 Park Hyun Woo

CONTEXT

A. Variable selection

B. Modeling (+ Data Partition)

C. Result of Modeling

VARIABLE SELECTION

DELETING OUTLIERS

```python
#데이터 삭제 open = 0인 데이터들
# 그리고 평균내서 store데이터에 추가 ~ 변수와 상관관계 파악하기 위해서
train.drop(train[train['Open']==0].index,inplace=True)
```

```python
train.reset_index(drop=True,inplace=True)

def find_outlier(data):
    Q1 , Q3 = np.percentile(data,[25,75])
    IQR = Q3 - Q1
    Over_outlier = Q3 + 1.5*IQR
    Low_outlier = Q1 - 1.5*IQR
    location = np.where((data>Over_outlier)|(data<Low_outlier))
    result = [list(location[0]),len(list(location[0]))]
    return result

locationOfOutlier  =  find_outlier(train['Sales'])[0]
numOfOutlier = find_outlier(train['Sales'])[1]

print("Number of Outlier is " , numOfOutlier)
print("percentage of Outlier of whole data ", (numOfOutlier/len(train))*100,"%")

train.drop(index = locationOfOutlier,inplace = True)
train
```

- I delete data of Open =0. Because this data not help predict sales information and don’t have another important information.
- And I delete outlier of Sales by using IQR.
- By using IQR 30000 data are deleted, this data is 3% of whole dataset

VARIABLE SELECTION IN STORE SET

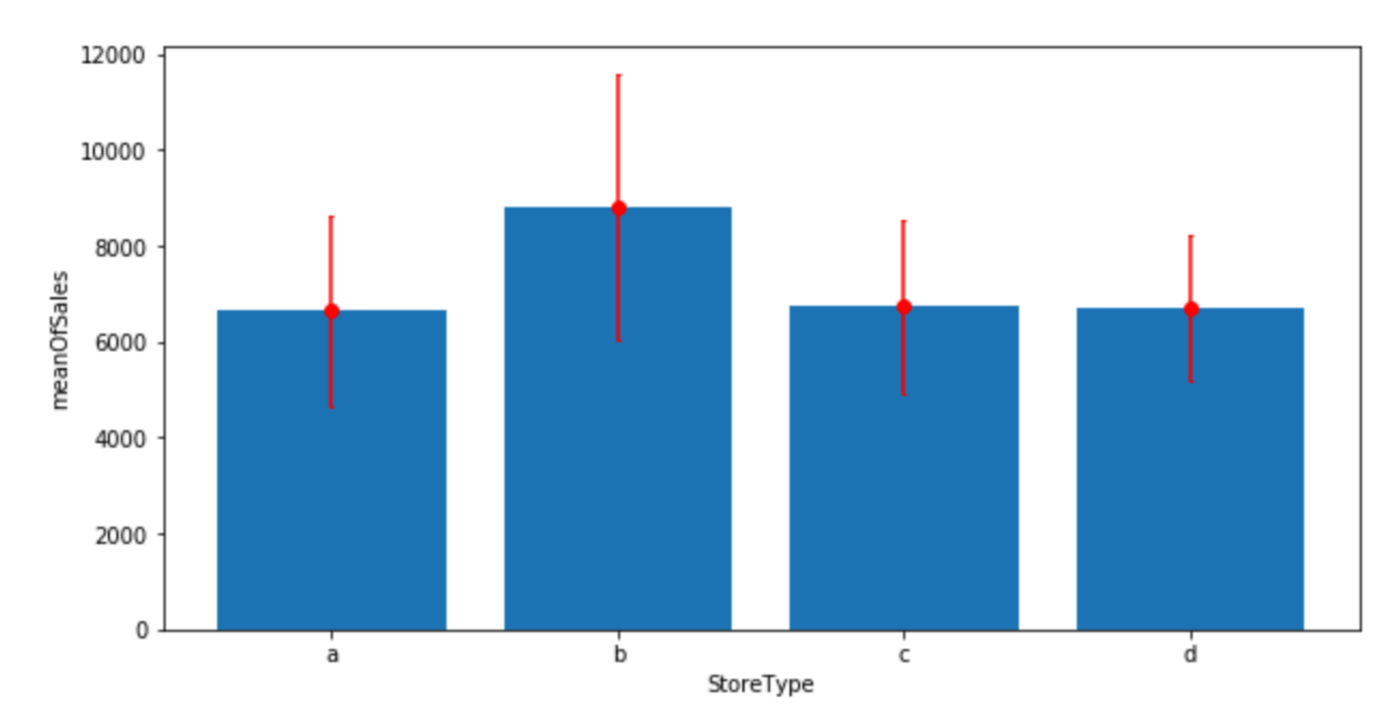
```python
# befor merge data s
# f-test & Data  for store data Assortment and StoreType
from scipy.stats import f_oneway
import matplotlib.pyplot as plt


sale_mean = train.groupby('Store')['Sales'].mean()
store['sale_mean'] = sale_mean.values

catvar = ['StoreType','Assortment','Promo2']

f_value ={}
for i in catvar:
    groups = [x[1].values for x in store.groupby([i])['sale_mean']]
    f_value[i] = f_oneway(*groups)
f_value
```

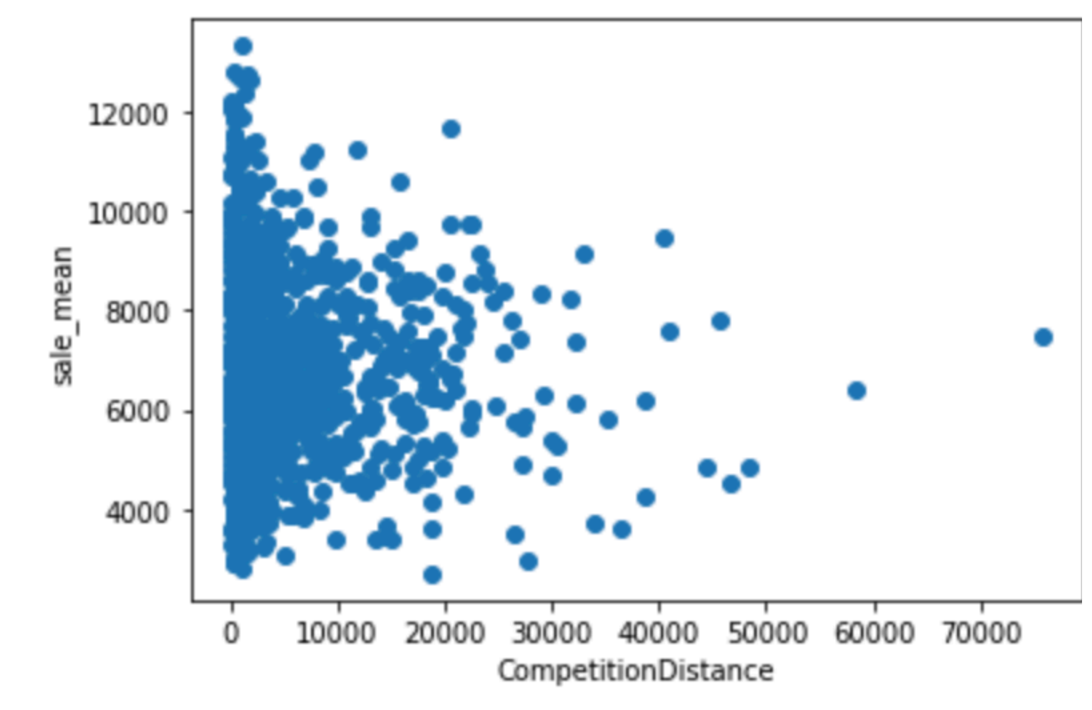
```
{'StoreType': F_onewayResult(statistic=7.489503916409775, pvalue=5.7746499655616875e-05),
 'Assortment': F_onewayResult(statistic=16.52706865516541, pvalue=8.452680077617934e-08),
 'Promo2': F_onewayResult(statistic=30.741717259621808, pvalue=3.679881968221907e-08)}
```

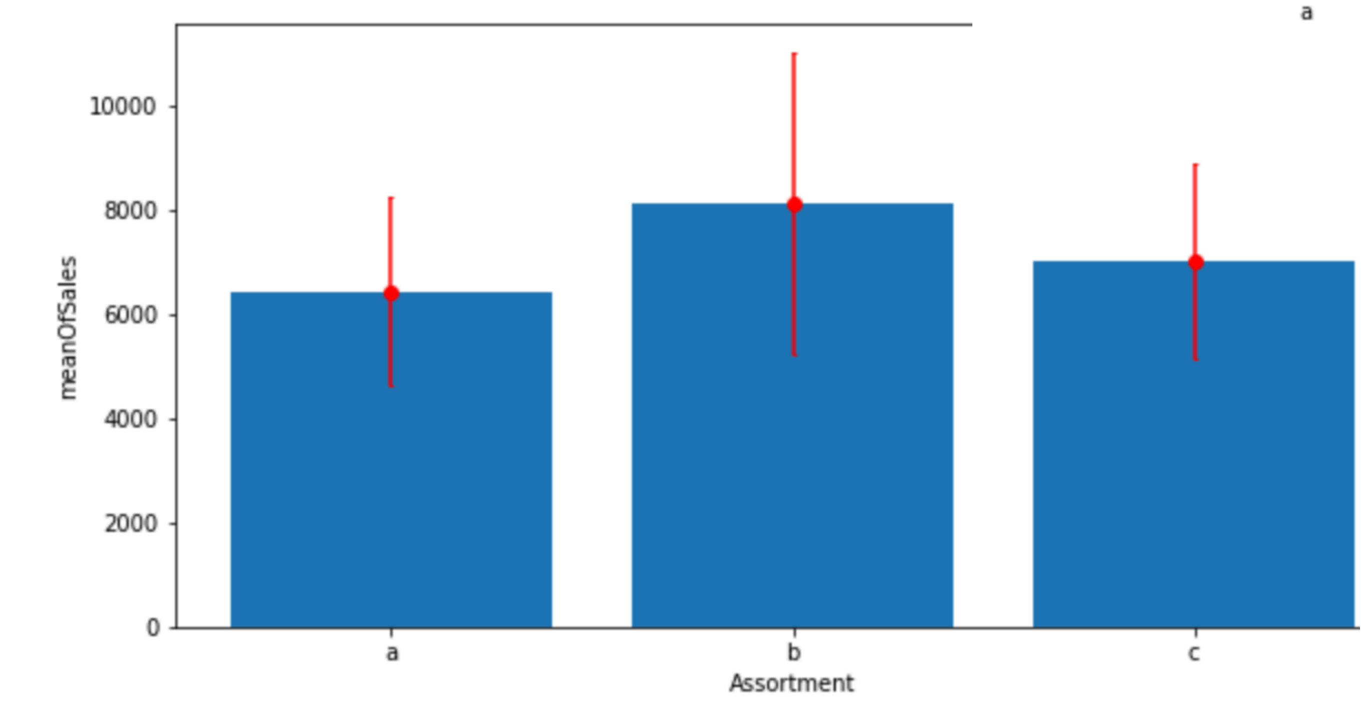
- Firstly I want to meaningful variables of Store data related to Sales.
- So I make new column in store dataset 'sale_mean'. This is mean of sales in each Store.
- And I think StoreType and Assortment and Promo2 is possible values in explanatory variables. So I take one-way Anova to sale_mean data .
- Those f-value all close to 0. This mean average of sale of (StoreType, Assortment, Promo2) differ

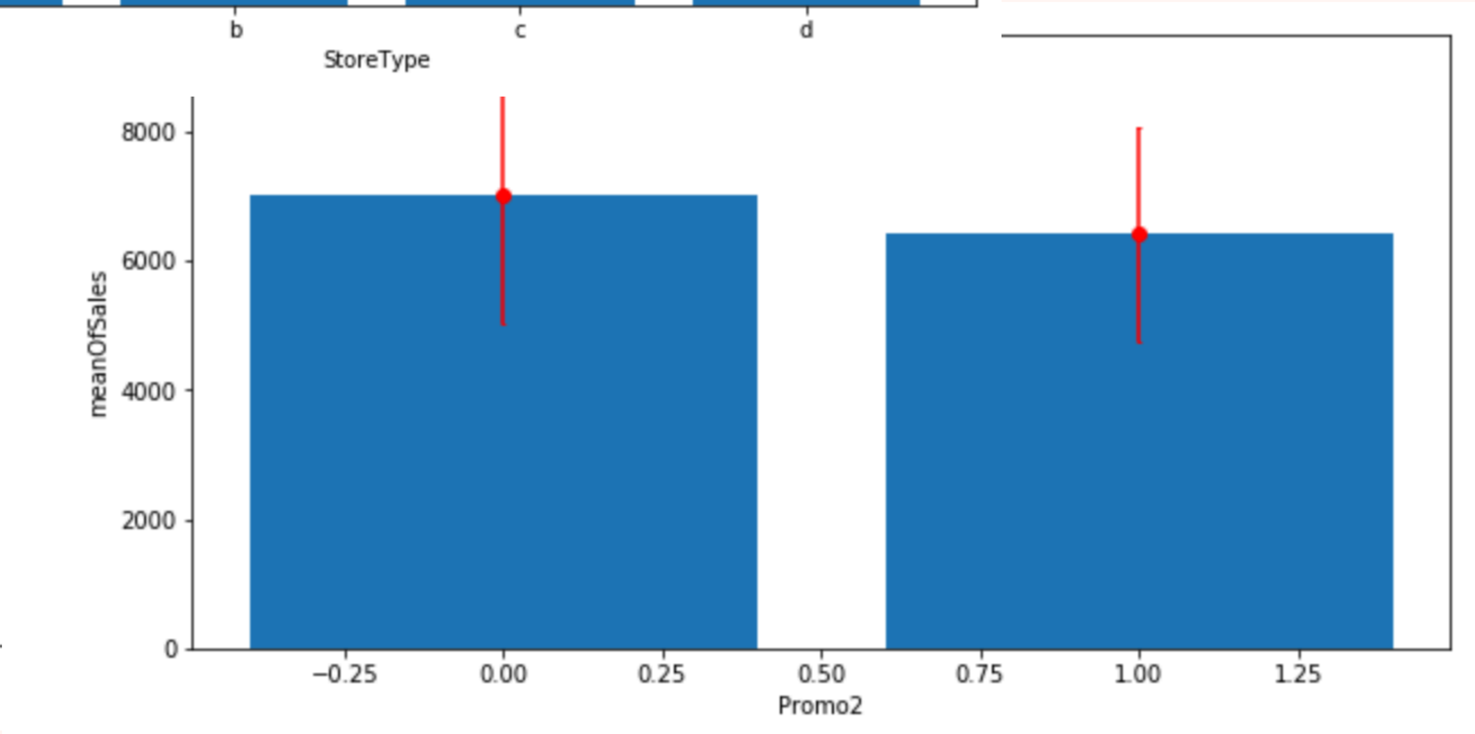
VARIABLE SELECTION IN STORE SET

```
plt.scatter(store['CompetitionDistance'], store['sale_mean'])
plt.xlabel('CompetitionDistance')
plt.ylabel('sale_mean')


for i in catvar:
    sale_mean = store.groupby(i)['sale_mean'].mean()
    sale_std = store.groupby(i)['sale_mean'].std()
    plt.figure(figsize=(10,5))
    plt.xlabel(i)
    plt.ylabel('meanOfSales')
    plt.bar(range(len(sale_mean)),sale_mean)
    plt.errorbar(sale_mean.index,sale_mean,yerr=sale_std, fmt='o', c='r',ecolor='r',capthic
```









- In graph about CompetitionDistance , around 0 competition value have many sale_mean values. And this can’t found regularity. So I don’t select CompetitionDistance variable.

- And in graph about another 3 categorical variables, meaningful difference exists in StoreType and Assortment. Promo2 don’t have meaningful difference

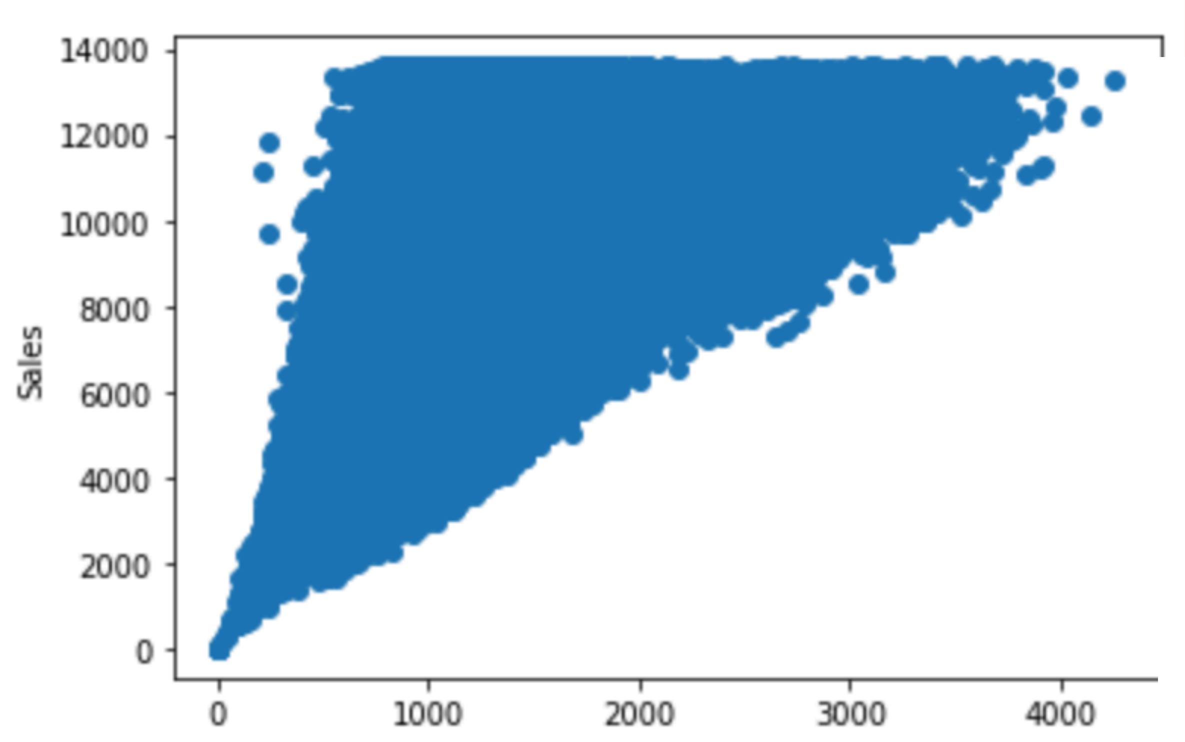
VARIABLE SELECTION IN TRAIN SET

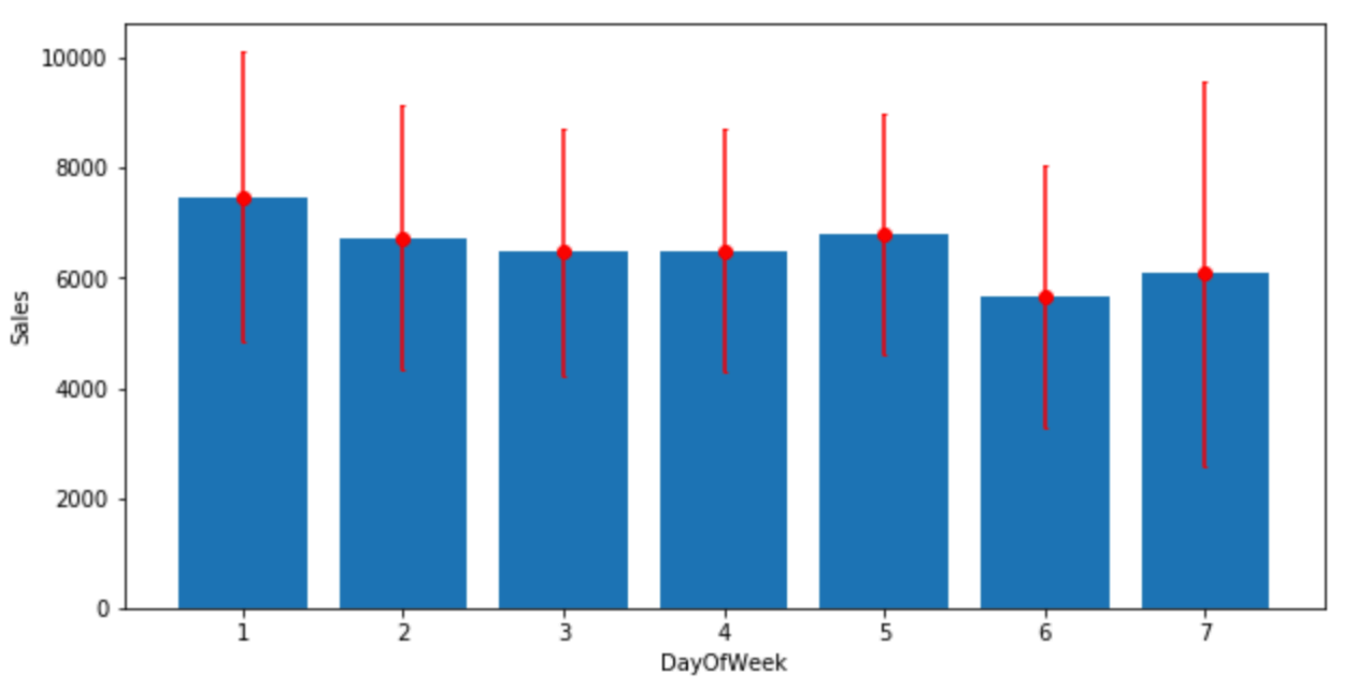
```
catvar =['StateHoliday','SchoolHoliday','Promo','DayOfWeek']


f_value ={}
for i in catvar:
    groups = [x[1].values for x in train.groupby([i])['Sales']]
    f_value[i] = f_oneway(*groups)
f_value
```

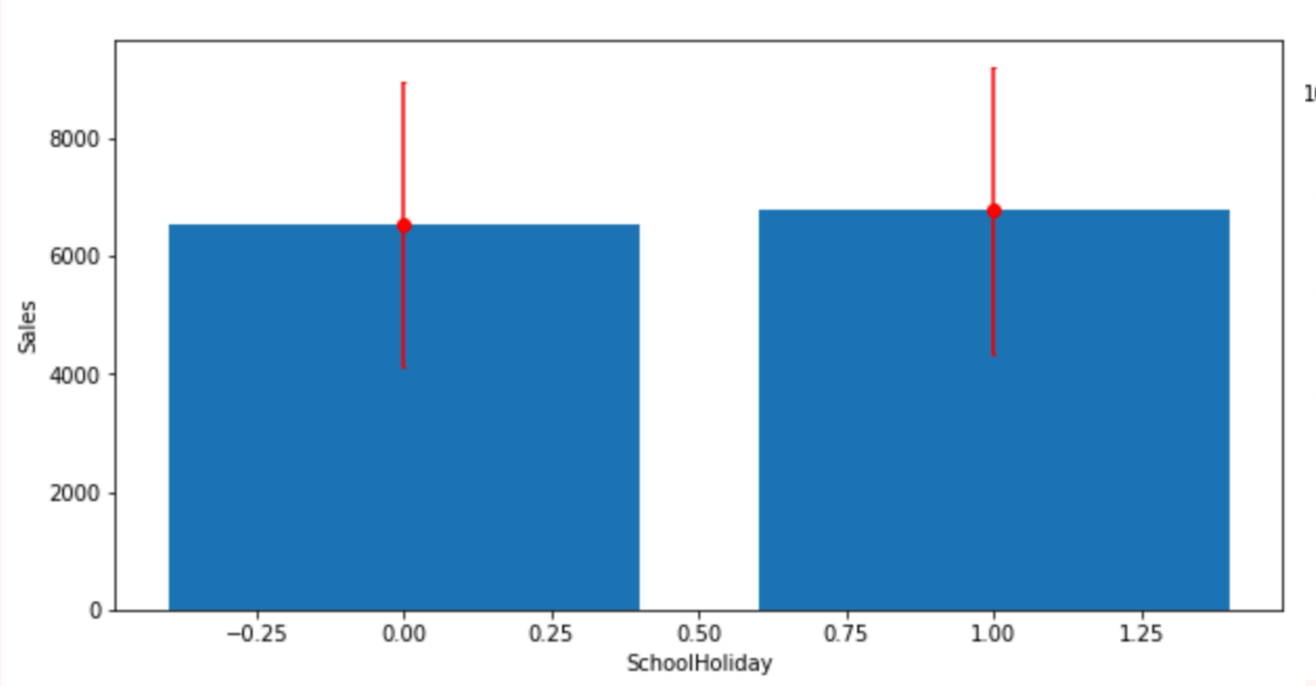
```
{'StateHoliday': F_onewayResult(statistic=3.76574141157973, pvalue=0.010222936253496073),
 'SchoolHoliday': F_onewayResult(statistic=1200.2217330785584, pvalue=8.501063011071299e-26
3),
 'Promo': F_onewayResult(statistic=157887.1072157874, pvalue=0.0),
 'DayOfWeek': F_onewayResult(statistic=7036.196758622229, pvalue=0.0)}
```

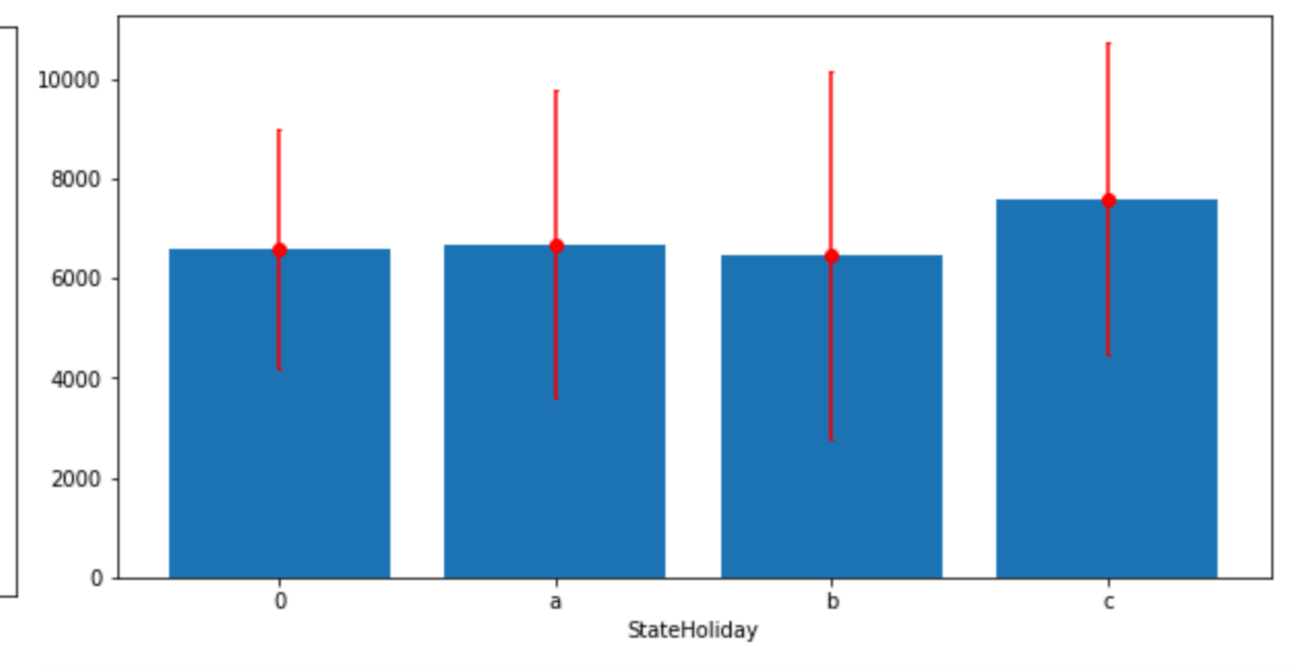
- Categorical variables which are meaningful are StateHoliday, SchoolHoliday, Promo and DayOfWeek.
- Those categorical variable's one-way Anova test's p-value close to 0.
- So mean of sales groupby them is differ each

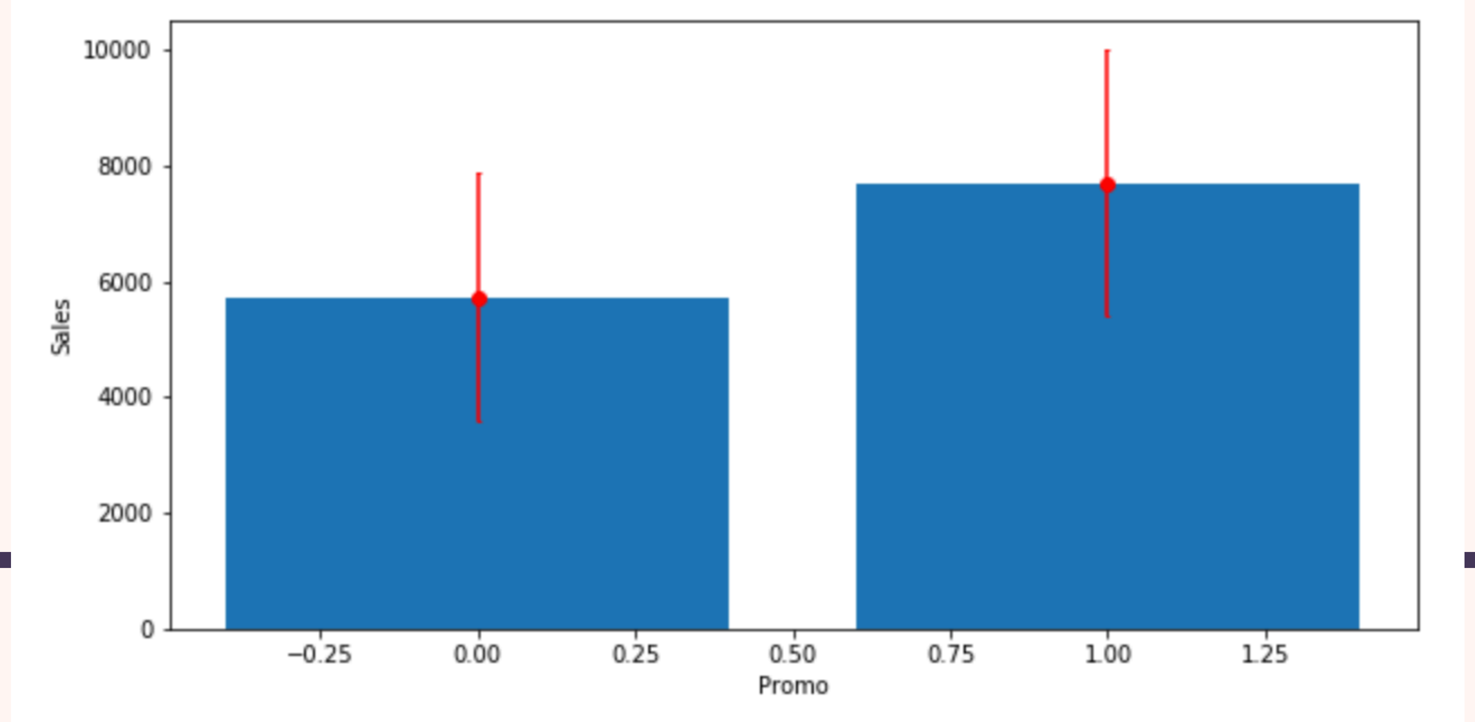
VARIABLE SELECTION IN TRAIN SET











- **Customer data is meaningful in common sense and those scatter plot shows that.**
- **Among 4 categorical variables, DayOfweek and Promo shows meaningful difference between data**
- **So I select customer DayOfWeek Promo variables to model**

MERGING TWO DATASET & MAKING DUMMY VARIABLES

```
store = store.drop(columns = ['CompetitionDistance','CompetitionOpenSinceMonth', 'Competitio
train = pd.merge(train,store, on = "Store")

catvar = ['StoreType','Assortment','Promo','DayOfWeek']

for c in catvar:
    dummy = pd.get_dummies(train[c],prefix = c, drop_first = True)
    train = pd.concat((train,dummy),axis=1)
```

```
train.columns
```

```
Index(['Store', 'DayOfWeek', 'Date', 'Sales', 'Customers', 'Open', 'Promo',
       'StateHoliday', 'SchoolHoliday', 'StoreType', 'Assortment', 'Promo2',
       'StoreType_b', 'StoreType_c', 'StoreType_d', 'Assortment_b',
       'Assortment_c', 'Promo_1', 'DayOfWeek_2', 'DayOfWeek_3', 'DayOfWeek_4',
       'DayOfWeek_5', 'DayOfWeek_6', 'DayOfWeek_7'],
      dtype='object')
```

- **Selected variables in store data set are Assortment and StoreType. So using merge method merge two datasets.**
- **After merge datasets, I make dummy variables for 4 categorical variables.**
- **Train data set have those columns in picture.**

MODELING

CHECK MULTICOLLINEARITY

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from patsy import dmatrices

y, X = dmatrices('Sales ~Customers+StoreType_b+StoreType_c+StoreType_d+Assortment_b+Assortmen
vif = pd.DataFrame()
vif["vif value"] = [variance_inflation_factor(X.values,i) for i in range(len(X.columns))]
vif["explanatory variables"] = X.columns
vif
```

	vif value	explanatory variables
0	17.356940	Intercept
1	1.364152	Customers
2	2.680438	StoreType_b
3	1.089902	StoreType_c
4	1.233840	StoreType_d
5	2.509553	Assortment_b
6	1.079218	Assortment_c
7	1.240741	Promo_1
8	1.744583	DayOfWeek_2
9	1.748424	DayOfWeek_3
10	1.715736	DayOfWeek_4
11	1.725435	DayOfWeek_5
12	1.968784	DayOfWeek_6
13	1.104302	DayOfWeek_7

- **This shows multicollinearity between explanatory variables.**
- **All of VIF values are lower than 10**
- **So this mean there are a little multicollinearity.**

DATA PARTITIONING

```
train = train.sort_values('Store')

X_train = train.loc[train['Date']<'2015-01-01']
X_train = X_train.drop(catvar+['Sales','Date','Store','SchoolHoliday','StateHoliday','Open',
X_train['intercept'] = 1


y_train = train.loc[train['Date']<'2015-01-01']
y_train = y_train['Sales']


X_vaild = train.loc[train['Date']>='2015-01-01']
X_vaild = X_vaild.drop(catvar+['Sales','Date','Store','SchoolHoliday','StateHoliday','Open',
X_vaild['intercept'] = 1


y_vaild = train.loc[train['Date']>='2015-01-01']
y_vaild = y_vaild['Sales']

X_train.reset_index(drop = True,inplace=True)
y_train.reset_index(drop = True,inplace=True)
X_vaild.reset_index(drop = True,inplace=True)
y_vaild.reset_index(drop = True,inplace=True)
```

- **For partitioning data, I make those code. Reference point is '2015-01-01'**
- **For training and testing, I make X_train and y_train**
- **For vaildating, I make X_vaild and y_vaild.**

BY OLS MODELING, CHECK MODEL SIGNIFICANCE AND VARIABLE SIGNIFICANCE

```python
import statsmodels.api as sm

model1 = sm.OLS(y_train,X_train)
result = model1.fit()
result.summary()
```

Dep. Variable:	Sales	**R-squared:**	0.780
Model:	OLS	**Adj. R-squared:**	0.780
Method:	Least Squares	**F-statistic:**	1.704e+05
Date:	Sun, 03 Oct 2021	**Prob (F-statistic):**	0.00
Time:	16:28:06	**Log-Likelihood:**	-5.2825e+06
No. Observations:	624629	**AIC:**	1.057e+07
Df Residuals:	624615	**BIC:**	1.057e+07
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Customers	6.9983	0.005	1286.834	0.000	6.988	7.009
StoreType_b	-2289.7321	19.560	-117.059	0.000	-2328.070	-2251.394
StoreType_c	-178.3845	4.404	-40.504	0.000	-187.016	-169.753
StoreType_d	1102.3546	3.461	318.495	0.000	1095.571	1109.138
Assortment_b	-4248.9158	24.283	-174.972	0.000	-4296.510	-4201.321
Assortment_c	319.6375	3.004	106.418	0.000	313.751	325.524
Promo_1	1177.5872	3.238	363.724	0.000	1171.242	1183.933
DayOfWeek_2	-350.7921	5.053	-69.427	0.000	-360.695	-340.889
DayOfWeek_3	-466.7062	5.073	-92.000	0.000	-476.649	-456.764
DayOfWeek_4	-538.7416	5.133	-104.963	0.000	-548.801	-528.682
DayOfWeek_5	-386.7261	5.088	-76.004	0.000	-396.699	-376.753
DayOfWeek_6	-114.8070	5.322	-21.574	0.000	-125.237	-104.377
DayOfWeek_7	-121.3058	24.790	-4.893	0.000	-169.894	-72.718
intercept	902.4699	5.979	150.947	0.000	890.752	914.188

- **Model's F-value close to 0. This mean this model have significance for predicting sales.**
- **And variables's t-test p-value all close to 0. This mean those variables have significance.**
- **And in error term, residual don't have normality.**

Omnibus:	32331.370	**Durbin-Watson:**	0.418
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	77318.492
Skew:	0.318	**Prob(JB):**	0.00
Kurtosis:	4.602	**Cond. No.**	1.60e+04

TWO REGRESSION MODEL

```python
from sklearn.model_selection import GroupKFold
from sklearn.linear_model import LinearRegression,Lasso

linear_model = LinearRegression()
lasso_model = Lasso(alpha = 1.0)

train_group = train.loc[train['Date']<'2015-01-01']
train_group = train_group['Store']

gkf = GroupKFold(n_splits=train_group.nunique())
linear_score = []
lasso_score = []

for train_index,test_index in gkf.split(X_train,y_train,np.array(train_group)):# y informati
    X1_train, X1_test= X_train.iloc[train_index], X_train.iloc[test_index]
    y1_train, y1_test= y_train.iloc[train_index], y_train.iloc[test_index]
    linear_model.fit(X1_train,y1_train)
    lasso_model.fit(X1_train,y1_train)
    linear_score.append(linear_model.score(X1_test,y1_test))
    lasso_score.append(lasso_model.score(X1_test,y1_test))
```

- I use LinearRegression Model and Lasso Model for predicting sales
- And I split X_train,y_train data by using group-K-Fold. Group is divided by store number.
- So I want to see data score in each store

RESULT

SCORE VALUE OF MODELS

```
LinearScore = list(filter(lambda x: x>-10, linear_score))
LassoScore = list(filter(lambda x: x>-10, lasso_score))

LinearScore  = pd.DataFrame(LinearScore,columns = ['ScoreOfLinearModel'])
LassoScore = pd.DataFrame(LassoScore,columns = ['ScoreOfLassoModel'])

print(LinearScore['ScoreOfLinearModel'].describe(),"\n\n\n")

print(LassoScore['ScoreOfLassoModel'].describe())

#LassoModel is a little better than LinearModel
```

```
count    1110.000000
mean        0.427847
std         0.883985
min        -8.990817
25%         0.358791
50%         0.694677
75%         0.821268
max         0.945245
Name: ScoreOfLinearModel, dtype: float64



count    1110.000000
mean        0.429338
std         0.876319
min        -8.959885
25%         0.359307
50%         0.694334
75%         0.821031
max         0.945974
Name: ScoreOfLassoModel, dtype: float64
```

- **First I delete 5 Score data because that have almost -100 score value**
- **And then those described datas shows score's data , over 50% score data are larger than 0.7 => this means over 50% store's sales prediction have 70% explanatory power**

MODEL COMPARING BY USING VALIDATIO NS

```python
linear_predict = linear_model.predict(X_vaild)
lasso_predict = lasso_model.predict(X_vaild)
```

```python
linear_predict = linear_model.predict(X_vaild)
lasso_predict = lasso_model.predict(X_vaild)

linear_predict = pd.DataFrame(linear_predict,columns = ['PredictSalesOfLinearModel'])
lasso_predict = pd.DataFrame(lasso_predict,columns = ['PredictSalesOfLassoModel'])

y_vaild = pd.DataFrame(y_vaild)
y_vaild = y_vaild.rename(columns = {'Sales':'RealSale'})
```

```python
result = pd.concat([linear_predict,lasso_predict,y_vaild],axis=1)
result
```

	PredictSalesOfLinearModel	PredictSalesOfLassoModel	RealSale
0	5398.972077	5416.937161	5263
1	6583.505141	6559.659844	5942
2	5088.182992	5103.588208	5289
3	4130.412770	4148.510641	4708
4	3607.487587	3626.002587	4042
...	...	...	...
188989	5370.891043	5374.074169	7295
188990	4786.967575	4791.442536	5570
188991	5025.783918	4990.336928	5025
188992	6618.584338	6619.377941	8509
188993	6357.459998	6361.308060	7345

188994 rows × 3 columns

```python
result['PredictSalesOfLinearModel'].describe()
```

```
count    188994.000000
mean       6506.344925
std        2050.252396
min       -1230.121286
25%        5084.188531
50%        6322.662595
75%        7717.364911
max       23761.879943
Name: PredictSalesOfLinearModel, dtype: float64
```

```python
result['PredictSalesOfLassoModel'].describe()
```

```
count    188994.000000
mean       6506.510730
std        2045.611151
min       -1111.294655
25%        5085.807175
50%        6322.638304
75%        7712.556811
max       23842.366877
Name: PredictSalesOfLassoModel, dtype: float64
```

```python
result['RealSale'].describe()
```

```
count    188994.000000
mean       6732.904605
std        2389.504938
min           0.000000
25%        4985.000000
50%        6419.000000
75%        8221.000000
max       13611.000000
Name: RealSale, dtype: float64
```

- **I make DataFrame for comparing two model.**
- **Above picture shows comparing predict data of Linear model and Lasso Model with sales data of validation set.**
- **The Lasso model is a little more similar with sales data of validation set than Linear Model. SO I can select Lasso model not Linear Model**

TRY TO PREDICT FUTURE_SALES BY USING VAR

```
data = data.diff().dropna()
# for stationary
```

```
forecast = VAR(data)
```

```
/opt/anaconda3/lib/python3.7/site-packages/statsmodels/tsa/base/tsa_model.py:218: ValueWarn
ing: A date index has been provided, but it has no associated frequency information and so
will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
/opt/anaconda3/lib/python3.7/site-packages/statsmodels/tsa/base/tsa_model.py:222: ValueWarn
ing: A date index has been provided, but it is not monotonic and so will be ignored when e.
g. forecasting.
  ' forecasting.', ValueWarning)
```

```
results = forecast.fit(2)
results.summary()
```

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Sun, 03, Oct, 2021
Time:                     17:49:30
--------------------------------------------------------------------
No. of Equations:         14.0000    BIC:                    88.4587
Nobs:                     780826.    HQIC:                   88.4543
Log likelihood:      -5.00439e+07    FPE:                2.59712e+38
AIC:                      88.4526    Det(Omega_mle):     2.59577e+38
--------------------------------------------------------------------
Results for equation Sales
==========================================================================
                  coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------
const              205.162002        18.047405           11.368           0.000
L1.Sales            -0.782762         0.003540         -221.092           0.000
L1.Customers        -0.156875         0.032541           -4.821           0.000
```

```
date = '2013-11-16'

pd.DataFrame(results.forecast(y = data[data.index == date].values,steps=10),
             index= range(1,11),columns=columnsOfData)
```

	Sales	Customers	StoreType_b	StoreType_c	StoreType_d	Assortment_b	Assortment_c	Promo_1	DayOfWeek_2
1	-2494.665589	-315.832995	-0.001490	0.002945	-0.060445	-0.000664	-0.095951	17.813035	12.559067
2	1464.428706	201.717182	0.005276	0.034603	0.102841	0.002674	0.116429	22.540482	64.777862
3	-1591.342115	-199.468953	0.006925	0.021690	0.067283	0.003995	0.085362	48.521138	133.720428
4	-2046.376416	-204.383527	0.006168	0.042900	0.082790	0.003663	0.099577	67.028203	5.042057
5	2813.928678	318.688741	0.009957	0.086806	0.142958	0.004790	0.173820	10.333819	10.430341
6	3.940656	-6.980003	0.009093	0.057307	0.098061	0.005192	0.116063	15.407140	10.334106
7	-76.554780	-11.078068	0.002038	0.020180	0.063654	0.001367	0.055175	15.411780	33.092116
8	-203.786152	-24.611557	0.002492	0.027892	0.057087	0.000990	0.063074	24.921014	26.975669
9	206.366141	16.589312	0.004648	0.030409	0.064399	0.002395	0.073125	25.284484	59.023289
10	-732.384087	-79.208079	0.006132	0.043977	0.086057	0.003172	0.102376	38.751365	69.274934

- I want to predict future_sales by using VAR.
- Right picture shows when int date = 2013-11-16, shows predicting values of all columns of model during 10 days(this steps = 10
- But the problem is the number of dummy variables are float.I can't know how to manage dummy variables(categorical variables) in that model. So I' cannot predict future sale

THANK YOU

PREDICTING BY VAR