MultipleRegression Final

January 12, 2024

Machine Learning Exercise 2 - Multiple Linear Regression

Ajay Badrinath

21011102020

[2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer as s
from sklearn.impute import KNNImputer as knn
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import mutual_info_regression
from sklearn.preprocessing import OrdinalEncoder
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
data = pd.read_csv(r'D:/house_pred.csv')
data_pred=pd.read_csv(r'D:/test.csv')
```

[349]:

```
class PreProcess():
    def __init__(self,data):
        self.data=data
        self.run()
    def run(self):
        self.ClearNull(threshold=0.5)
        l=self.get_all_Null(dtype='float64')
        self.knn_impute(2,l)
        #self.outlier_remove()
        self.data=self.data.dropna()
        self.one_hot_encoding()
        self.StdScale()
        self.outlier_remove('SalePrice')
        self.drop_correlation()

        self.drop_vif(thresh=4.5)
    def drop_correlation(self):
        k=Utils_Suite(self.data).compute_correlation(0.3)
        f=pd.DataFrame(k)
        m=list(f[(f['SalePrice']<0.1) & (f['SalePrice']>-0.1)].index)
        self.data=self.data.drop(columns=m)

    def ClearNull(self,threshold):
        x=self.data.isna().sum()>0

        for i in  list(x.index):
            thresh=self.data[i].isna().sum()/len(self.data)
            if(x[i]==True and thresh>threshold):
                print(i,self.data[i].isna().sum())
                self.data=self.data.drop(i,axis=1)


    def knn_impute(self,n_neighbors,col_list):
        imputer=knn(n_neighbors=n_neighbors)
        for i in col_list:
            self.data[i]=imputer.fit_transform(self.data[[i]])[0][0]

    def arbitrary_remove(self):
        #data=data.drop(columns=['LotFrontage','MasVnrArea','GarageYrBlt'])
        self.data=self.data.drop('Id',axis=1)

    def get_all_Null(self,dtype=""):
        x=self.data.isna().sum()>0
        l=[]
        for i in  list(x.index):
            thresh=self.data[i].isna().sum()/len(self.data)
            if(x[i]==True and (data[i].dtypes==dtype) ):
                print(i,data[i].isna().sum())
                l+=[i]
        return l

    def outlier_remove(self,col):

        q1=self.data[col].quantile(0.25)
        q3=self.data[col].quantile(0.75)
        iqr=q3-q1
        l_whis=q1-1.5*iqr
        u_whis=q3+1.5*iqr
        self.data= self.data[(self.data[col]>=l_whis)& (self.data[col]<=u_whis)]

    #Depricated  ....
    def outlier_remove_deprecated(self):
        for col in self.data.columns:
            if self.data[col].dtypes!='object':
                q1=self.data[col].quantile(0.25)
                q3=self.data[col].quantile(0.75)
                iqr=q3-q1
                l_whis=q1-1.5*iqr
                u_whis=q3+1.5*iqr
                self.data= self.data[(self.data[col]>=l_whis)& (self.
↪data[col]<=u_whis)]
        return self.data

    def one_hot_encoding(self):
        z=(self.data.dtypes=='object')
        k=pd.DataFrame(z)
        obj_list=list(k[k[0]==True].index)
        print(obj_list)
        for i in obj_list:
            dummy=pd.get_dummies(self.data[i],prefix=i,drop_first=True)
            #print(dummy)
            self.data=self.data.drop(i,axis=1)
            self.data=self.data.join(dummy)
            #self.data=pd.concat([self.data,dummy],axis=1)


    def StdScale(self):
        for i in self.data.columns:
            if self.data[i].dtypes!='object' and i!='SalePrice':
                scale = StandardScaler().fit(self.data[[i]])

                self.data[i] = scale.transform(self.data[[i]])


    ## DANGER ZONE Col Spare NEEDED To Keep y_pred.
    def drop_vif(self,thresh=5,col_Spare=['SalePrice','intercept']):


        vif=Utils_Suite(self.data).compute_vif()
        z1=vif[vif["vif"]>thresh]
        z1=z1.sort_values(by='vif', kind='mergesort',ascending=[False])
        while True:
            try:
                col=z1.iloc[0,0]
                if z1.empty:
                    break
                if col in col_Spare:
                    z1=z1.iloc[1:]
                    continue
                self.data=self.data.drop(col,axis=1)
                vif=Utils_Suite(self.data).compute_vif()
                z1=vif[vif["vif"]>thresh]
                z1=z1.sort_values(by='vif', kind='mergesort',ascending=[False])
            except IndexError:
                break


    def write_df(self):
        return self.data
```

[351]:

```
class Utils_Suite():
    def __init__(self,data):
        self.data=data
    def compute_correlation(self,threshold=0.3):
        matrix=self.data.corr(numeric_only=True)
       ␣
 ↪x=matrix[(matrix["SalePrice"]<threshold)&(matrix["SalePrice"]>-threshold)]["SalePrice"]
        return x
    def compute_mutual_information(self,thresh=0.1):
        enc = OrdinalEncoder()
        df_encoded = enc.fit_transform(self.data)
        mi_scores = mutual_info_regression(df_encoded, self.data['SalePrice'])
        mi_scores_df = pd.DataFrame(mi_scores, index=self.data.columns,␣
 ↪columns=['Score'])
        return mi_scores_df[mi_scores_df['Score']<thresh]
    def compute_vif(self):
        x=self.data.iloc[:,:-1]
        y=self.data.iloc[:,-1]
        x=pd.DataFrame(x)

        x['intercept']=1
        vif=pd.DataFrame()
        vif['variable']=x.columns
        vif['vif']=[variance_inflation_factor(x.values,i)for i in range(x.
 ↪shape[1])]
        return vif
```

[352]:

```
class Model():
    def __init__(self,x_train,y_train,x_test,y_test):
        self.x_train=x_train
        self.x_test=x_test
        self.y_train=y_train
        self.y_test=y_test
        self.y_pred=0

    def fit(self):
            self.reg = LinearRegression()
            self.reg.fit(self.x_train,self.y_train)
            return self.reg
    def predict(self):
         self.y_pred=self.reg.predict(self.x_test)
         return self.y_pred
    def score_metric(self):
         return r2_score(self.y_test,self.y_pred)
```

[353]:

```
k=PreProcess(data=data)
```

```
Alley 1369
PoolQC 1453
Fence 1179
MiscFeature 1406
LotFrontage 259
MasVnrArea 8
GarageYrBlt 81
['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd',
'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC',
'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu',
'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive',
'SaleType', 'SaleCondition']
```

```
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
```

```
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
```

```
  return 1 - self.ssr/self.centered_tss
```

```
[354]: data=k.write_df()
```

```
[355]: col=list(data.columns)
       col.remove('SalePrice')
       col.append('SalePrice')
       data=data[col]
```

```
[368]: x=data.iloc[:,:-1]
       y=data.iloc[:,-1]
       x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
        ↪3,random_state=2024)
```

```
[357]: u=Utils_Suite(data).compute_vif()
```

```
e:\anaconda\lib\site-packages\statsmodels\regression\linear_model.py:1752:
RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
```

```
[358]: u[u['vif']>4]
```

```
[358]:                  variable       vif
       6             BsmtFinSF1  4.060141
       78           SaleType_WD  4.073770
       80  SaleCondition_Partial  4.841633
```

```
[369]: MR_Model=Model(x_train,y_train,x_test,y_test)
       reg=MR_Model.fit()
```

```
[370]: reg.score(x,y)
```

```
[370]: 0.7909355723036731
```

```
[373]: y_pred=MR_Model.predict()
```

```
[372]: MR_Model.score_metric()
```

```
[372]: 0.689393821581118
```

```
[376]: from sklearn.metrics import mean_squared_error
       np.sqrt(mean_squared_error(y_test,y_pred))
```

```
[376]: 36857.56429672354
```

```
[ ]:
```