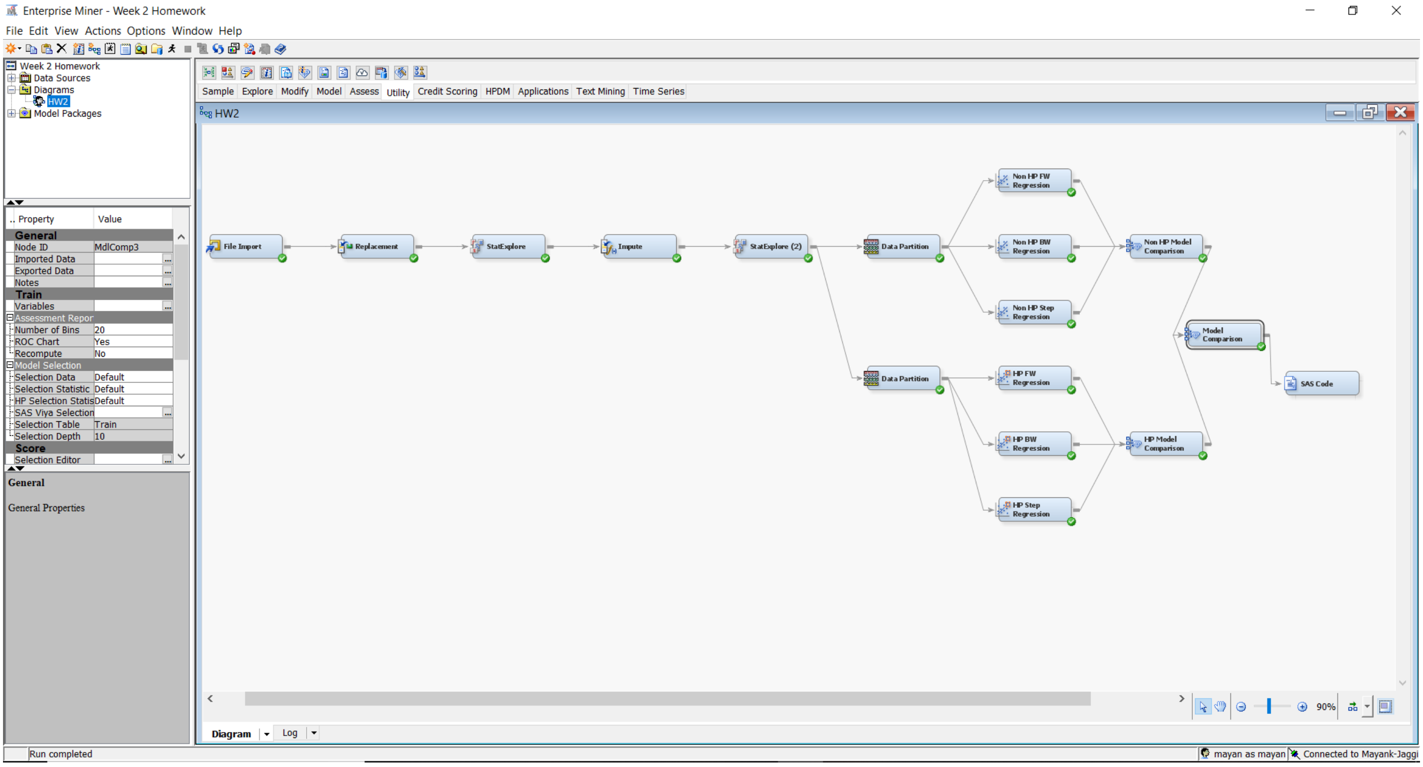
STAT 656
Homework 2

Name-Mayank Jaggi

UIN-526005299

PART 1: SAS ENTERPRISE MINER

Screenshot of Project Window



.. Property	Value
General	
Node ID	MdlComp3
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Assessment Repor	
Number of Bins	20
ROC Chart	Yes
Recompute	No
Model Selection	
Selection Data	Default
Selection Statistic	Default
HP Selection Statis	Default
SAS Viya Selection	...
Selection Table	Train
Selection Depth	10
Score	
Selection Editor	...

General

General Properties

Note: Couldn't complete second and third part of Part 1

PART 2: PYTHON
PYTHON PROGRAM

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 30 12:29:26 2019

@author: mayank
"""

import pandas as pd
import numpy as np
from AdvancedAnalytics import ReplaceImputeEncode
from sklearn import preprocessing
from sklearn.model_selection import train_test_split

df1 = pd.read_excel("diamondswmissing.xlsx")     #data file name
df2 = df1.dropna(subset = ['price'])  # removing the column object as its the target


print(df2)

#Missing values and outliers

data_map = {\
            'obs'    : [2,(1,53940),[0,0]], \
            'Carat'  :[0,(0.2,5.5),[0,0]], \
            'cut'    :[2,('Fair','Good', 'Ideal','Premium','Very Good'), [0,0]],\
            'color'  :[2,('D','E','F','F','H','I','J'), [0,0]],\
            'clarity':[2,('I1','IF','Sl1','Sl2','VS1','VS2','VVS1', 'VVS2'),[0]],\
            'depth'  :[0,(40,80),[0,0]],\
            'table'  :[0,(40,100),[0,0]],\
            'x'      :[0,(0,11),[0,0]],\
            'y'      :[0,(0,60),[0,0]],\
            'z'      :[0,(0,32),[0,0]],\
            }

# 0 for Interval and 2 for Nominal

rie = ReplaceImputeEncode(data_map=data_map, display=True)
df_rie = rie.fit_transform(df2)


#Imputing Missing Values

interval_att=['Carat','depth','table','x','y','z']      # list of attributes with
interval data type
interval_data=df2.as_matrix(columns=interval_att)
interval_impute=preprocessing.Imputer(strategy='mean')
interval_data_imputed = interval_impute.fit_transform(interval_data)
```

```
print("Imputed Interval Data:\n", interval_data_imputed)

map_cut={'Fair':1,'Good':2,'Ideal':3,'Premium':4,'Very Good':5}
map_color={'D':1,'E':2,'F':3,'G':4,'H':5,'I':6,'J':7}
map_clarity={'I1':1,'IF':2,'Sl1':3,'Sl2':4,'VS1':5,'VS2':6,'VVS1':7, 'VVS2':8}

df2['cut']=df2['cut'].map(map_cut)
df2['color']=df2['color'].map(map_color)
df2['clarity']=df2['clarity'].map(map_clarity)

nominal_att = ['cut','color','clarity']              # list of attributes with
nominal data type
nominal_data = df2.as_matrix(columns=nominal_att)

cat_impute = preprocessing.Imputer(strategy='most_frequent')
nominal_data_imputed = cat_impute.fit_transform(nominal_data)


# Adding imputed data in the data frame

df2[['cut','color','clarity']] = nominal_data_imputed
df2[['Carat','depth','table','x','y','z']] = interval_data_imputed
df2.head()


#Encoding

scaler = preprocessing.StandardScaler()             # create instance of
standardscaler()
scaler.fit(interval_data_imputed)
interval_data_scaled = scaler.transform(interval_data_imputed)
print("Imputed & Scaled Interval Data\n", interval_data_scaled)

# Create instances of OneHotEncoder & Selecting Attributes
one_hot = preprocessing.OneHotEncoder()
hot_array = one_hot.fit_transform(nominal_data_imputed).toarray()
print(hot_array)
print(df2)

from pandas import ExcelWriter
writer_file = ExcelWriter('Python_Export.xlsx')
df2.to_excel(writer_file)
writer_file.save()


from AdvancedAnalytics import linreg
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression

y = df2['price']
```

```
x = df2.drop('price',axis=1)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=1)

lr=LinearRegression()

col=[]
for i in range(x_train.shape[1]):
    col.append('X'+str(i))

lr.fit(x_train,y_train)
print("\n*** LINEAR REGRESSION ***")
linreg.display_coef(lr, x_train, y_train, col)
linreg.display_metrics(lr, x_train, y_train)

y_hat= lr.predict(x_test)
xtest1 = np.asanyarray(x_test)
ytest1 = np.asanyarray(y_test)

# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % lr.score(xtest1, ytest1))

print("Residual sum of squares: %.2f"
      % np.mean((y_hat - y_test) ** 2))



#Mean, max and min of predicted value
pred_mean = y_hat.mean(axis = 0)
print("\nPredicted mean\n",pred_mean)

pred_max = y_hat.argmax(axis = 0)
print("\nPredicted maximum\n",pred_max)

pred_min = y_hat.argmin(axis = 0)
print("\nPredicted minimum\n",pred_min)


#Mean, max and min of actual value
actual_mean = y_test.mean(axis = 0)
print("\nActual mean\n",actual_mean)

actual_max = y_test.idxmax(axis = 0)
print("\nActual maximum\n",actual_max)

actual_min = y_test.idxmin(axis = 0)
print("\nActual minimum\n",actual_min)

print("\nFirst 15 predicted values\n",y_hat[0:14])
```

```
OUTPUT

Predicted mean
 3951.0402142121225

Predicted maximum
 7912

Predicted minimum
 3747

Actual mean
 3900.195464095909

Actual maximum
 27746

Actual minimum
 2

First 15 predicted values
 [ -304.44466026  6528.1168117   3540.81803326  -405.33512559
  7850.36334435  2374.06773597  7145.61582818   394.55316417
  9704.89420857  1069.2609436     31.02330734  3114.24439391
  3512.01940475 -1152.66130589]
```