jupyter diamondSalesReg Last Checkpoint: Last Sunday at 2:32 PM (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel)

Run Code

In [ ]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import pickle
import joblib
```

Reading the dataset

In [ ]:
```
data = pd.read_csv("diamonds.csv")
data.head()
```

In [ ]:
```
data.info()
```

Data Preprocessing

In [ ]:
```
#checking for null values in the dataset
data.isnull().sum()
```

In [ ]:
```
data = data.drop(columns='Unnamed: 0')
```

In [ ]:
```
data.head()
```

In [ ]:
```
data.nunique()
```

In [ ]:
```
#Deleting any duplicated rows

data[data.duplicated()]
```

In [ ]:
```
data = data.drop_duplicates()
```

jupyter diamondSalesReg Last Checkpoint: Last Sunday at 2:32 PM (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (pykernel)

Run Code

219566	Radiant	F	SI1	10.02	unknown	GIA	Very Good	Excellent	Yes	N	...	7.76	unknown	unknown	unkno

3880 rows × 25 columns

In [9]:
```
data = data.drop_duplicates()
```

In [10]:
```
data[data.duplicated()]
```

Out[10]:

	cut	color	clarity	carat_weight	cut_quality	lab	symmetry	polish	eye_clean	culet_size	...	meas_depth	girdle_min	girdle_max	fluor_color	fluor_intensit

0 rows × 25 columns

In [ ]:
```
# Changing categorical values by encoding using Label encoder
from sklearn.preprocessing import LabelEncoder
```

In [ ]:
```
lnc = LabelEncoder()
df = data.copy()
for i in data:
    r = data[i].dtypes
    if r == 'object':
        df[i] = lnc.fit_transform(data[i])
        joblib.dump(lnc,i+'.joblib',compress=9)
```

In [ ]:
```
df.head()
df.info()
```

Outlier detection

In [ ]:
```
#we are gonna use plotting to understand the dataset

for i in df:
    r = df[i].dtypes
    if r == 'float64':
        plt.scatter(range(len(df[i])), df[i],label=i)
        plt.legend()
```

```
 23  fancy_color_intensity      215823 non-null  int64
 24  total_sales_price          215823 non-null  int64
dtypes: float64(6), int64(19)
memory usage: 42.8 MB
```

Outlier detection

In [ ]:
```
#we are gonna use plotting to understand the dataset

for i in df:
    r = df[i].dtypes
    if r == 'float64':
        plt.scatter(range(len(df[i])), df[i],label=i)
        plt.legend()
        plt.show()
```

In [ ]:
```
df = df[(df.meas_length < 40)]
df = df[(df.meas_width < 30)]
df = df[(df.meas_depth < 30)]
```

In [ ]:
```
#checking if measured parameters are 0 or not

df[(df.meas_width == 0) | (df.meas_depth == 0) | (df.meas_length == 0) | (df.depth_percent == 0) | (df.table_percent
```

In [ ]:
```
#since it is not substantial we can remove this and store in a different DF to check for model accuracy
df2 = df.copy()
df2 = df2[(df2.meas_width != 0)]
df2 = df2[(df2.meas_depth != 0)]
df2 = df2[(df2.meas_length != 0)]
df2 = df2[(df2.depth_percent != 0) ]
df2 = df2[(df2.table_percent != 0)]

print(df.shape)
print(df2.shape)
```

jupyter diamondSalesReg Last Checkpoint: Last Sunday at 2:32 PM (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (pykernel)

Run Code

44	10	0	7	0.11	0	2	4	0	4	8	...	0.00	9	9	5
45	10	1	7	0.11	0	2	4	4	4	8	...	0.00	9	9	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
218830	10	1	4	5.07	0	0	0	4	4	3	...	0.00	9	9	0
219389	7	10	10	15.52	5	0	4	0	4	8	...	6.34	1	3	5
219415	10	2	7	6.28	0	0	0	0	4	3	...	7.24	4	4	5
219530	7	4	7	11.95	5	1	4	0	4	8	...	7.76	1	1	5
219570	10	0	3	5.34	4	0	4	4	4	8	...	6.95	9	9	5

5086 rows × 26 columns

In [ ]:
```
#since it is not substantial we can remove this and store in a different DF to check for model accuracy
df2 = df.copy()
df2 = df2[(df2.meas_width != 0)]
df2 = df2[(df2.meas_depth != 0)]
df2 = df2[(df2.meas_length != 0)]
df2 = df2[(df2.depth_percent != 0) ]
df2 = df2[(df2.table_percent != 0)]

print(df.shape)
print(df2.shape)
```

Feature selection and Data splitting

In [ ]:
```
#lets do a quantitative analysis by feature correlation

from sklearn.preprocessing import StandardScaler, MinMaxScaler
ss = MinMaxScaler()
# done to normalize inputs


corr = df2.corr()
sns.heatmap(corr)
```

jupyter diamondSalesReg Last Checkpoint: Last Sunday at 2:32 PM (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (pykernel)

Run Code

```
print(df.shape)
print(df2.shape)
```

```
(215599, 25)
(210513, 25)
```

Feature selection and Data splitting

In [ ]:
```
#lets do a quantitative analysis by feature correlation

from sklearn.preprocessing import StandardScaler, MinMaxScaler
ss = MinMaxScaler()
# done to normalize inputs


corr = df2.corr()
sns.heatmap(corr)
```

In [ ]:
```
#lets first define all features
features = []
for i in df2:
    if i != 'total_sales_price':
        features.append(i)
print(features)
```

In [ ]:
```
X = df2[features]
y = df2['total_sales_price']

print(X.shape)
print(y.shape)
'''
#miniX = np.min(X)
#maxiX = np.max(X)
miniy = np.min(y)
maxiy = np.max(y)
#print(miniX)
print(maxiX)
```

jupyter diamondSalesReg Last Checkpoint: Last Sunday at 2:32 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
y = df2['total_sales_price']

print(X.shape)
print(y.shape)
'''
#miniX = np.min(X)
#maxiX = np.max(X)
miniy = np.min(y)
maxiy = np.max(y)
#print(miniX)
print(maxiX)
#X = (X-miniX)/(maxiX - miniX)
y = (y-miniy)/(maxiy - miniy)
'''
#X = ss.fit_transform(X)
#y = ss.fit_transform(y.values.reshape(-1,1))
print(X.shape)
print(y.shape)
```

```
(210513, 24)
(210513,)
(210513, 24)
(210513,)
```

In [ ]:
```
#using f_regression score we can slect k best features
from sklearn.feature_selection import SelectKBest as skb
from sklearn.feature_selection import f_regression

#first we analyze all scores

fs = skb(f_regression,k='all')
fs.fit(X,y)
for i in range(len(fs.scores_)):
    print('feature %d: %f' %(i,fs.scores_[i]))
```

In [ ]:
```
#now we select 15 best features out of 24 and get their names

fs = skb(f_regression, k = 15)
Xnew = fs.fit_transform(X,y)
f = np.array(features)
```

jupyter diamondSalesReg Last Checkpoint: Last Sunday at 2:32 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (pykernel)

Run Code

```
(210513, 24)
(210513,)
```

In [ ]:
```
#using f_regression score we can slect k best features
from sklearn.feature_selection import SelectKBest as skb
from sklearn.feature_selection import f_regression

#first we analyze all scores

fs = skb(f_regression,k='all')
fs.fit(X,y)
for i in range(len(fs.scores_)):
    print('Feature %d: %f' %(i,fs.scores_[i]))
```

In [ ]:
```
#now we select 15 best features out of 24 and get their names

fs = skb(f_regression, k = 15)
Xnew = fs.fit_transform(X,y)
f = np.array(features)
filt = fs.get_support()
f = f[filt]
print(f,len(f))
```

In [ ]:
```
Xnew = X
y = y.values.reshape(-1,1)
print(Xnew.shape)
print(y.shape)
```

In [ ]:
```
#final check
print(Xnew,Xnew.shape)
print(y,y.shape)
```

In [ ]:
```
#splitting the dataset into training and testing
from sklearn.model_selection import train_test_split as tts

X_train, X_test, y_train, y_test = tts(Xnew, y, test_size = 0.2, random_state = 40)
```

jupyter diamondSalesReg Last Checkpoint: Last Sunday at 2:32 PM (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (pykernel)

Run Code

In [25]:
```
#splitting the dataset into training and testing
from sklearn.model_selection import train_test_split as tts

X_train, X_test, y_train, y_test = tts(Xnew, y, test_size = 0.2, random_state = 40)

print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(168410, 24) (168410, 1)
(42103, 24) (42103, 1)
```

Modelling the regression

In [ ]:
```
#for this we can use the inbuilt LinearRegression model in sklearn
from sklearn.linear_model import LinearRegression

mlr = LinearRegression()

mlr.fit(X_train, y_train)

print("Intercept: ",mlr.intercept_)
print("Coeffs: ", list(zip(X, mlr.coef_)))
```

In [ ]:
```
y_pred_mlr = mlr.predict(X_test)
```

In [ ]:
```
#metrics

from sklearn import metrics

MABE = metrics.mean_absolute_error(y_test, y_pred_mlr)
r2 = mlr.score(X_test,y_test)*100

print("R squared score: ",r2)
print("Mean absolute error: ",MABE)
```

jupyter diamondSalesReg Last Checkpoint: Last Sunday at 2:32 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
print("R squared score: ",r2)
print("Mean absolute error: ",MABE)
```

Model number 2: ANN

In [ ]:
```
import tensorflow
from keras.models import Sequential
from keras.layers import Dense
```

In [ ]:
```
model = Sequential()
model.add(Dense(units = 100, input_dim = 24, kernel_initializer = 'normal', activation='relu'))
model.add(Dense(units = 50, kernel_initializer = 'normal', activation='relu'))
model.add(Dense(1, kernel_initializer = 'normal'))

#Since it is a curve fitting problem loss is MSE
#Lets add a history to track the loss
model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(X_train,y_train, validation_split = 0.2, batch_size = 2000, epochs = 50, verbose = 1 )
```

In [ ]:
```
#plotting loss curve
'''
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title("Model Accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title("Model loss")
plt.ylabel("Loss")
plt.xlabel("Epoch")
```

jupyter diamondSalesReg Last Checkpoint: Last Sunday at 2:32 PM (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (pykernel)

Run Code

```
68/68 [==============================] - 0s 6ms/step - loss: 204919076.5217 - val_loss: 164007776.0000
Epoch 49/50
68/68 [==============================] - 0s 6ms/step - loss: 192203196.5217 - val_loss: 163966288.0000
Epoch 50/50
68/68 [==============================] - 0s 6ms/step - loss: 216573136.9275 - val_loss: 162738304.0000
```

In [ ]:
```
#plotting loss curve
'''
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title("Model Accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title("Model loss")
plt.ylabel("Loss")
plt.xlabel("Epoch")
plt.show()
'''

pd.DataFrame(history.history).plot(figsize = (10,10))
plt.show()
```

In [ ]:
```
#predictions

pred = model.predict(X_test)
#pred_denorm = pred*(maxiy-miniy) + miniy
#y_test_denorm = y_test*(maxiy-miniy) + miniy


from sklearn.metrics import r2_score
r2 = r2_score(y_test, pred)*100

print("R Squared: ",r2)
```

In [ ]:
```
#plotting the fit
```

jupyter diamondSalesReg Last Checkpoint: Last Sunday at 2:32 PM (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (pykernel)

Run Code

In [29]:
```
#predictions

pred = model.predict(X_test)
#pred_denorm = pred*(maxiy-miniy) + miniy
#y_test_denorm = y_test*(maxiy-miniy) + miniy


from sklearn.metrics import r2_score
r2 = r2_score(y_test, pred)*100

print("R Squared: ",r2)
```

```
2023-04-28 15:24:18.200026: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:112] Plugin optimizer for device_type GPU is enabled.

R Squared:  74.4120433938284
```

In [ ]:
```
#plotting the fit

print(pred[5024])
print(y_test[5024])
plt.figure(figsize=(50,10))
plt.plot(y_test[5000:5100],'ro--',label='actual')
plt.plot(pred[5000:5100], 'b+-', label='predicted')
plt.legend()
plt.show()
```

Saving Model

In [ ]:
```
model.save('tf_m_1.0.0.h5')
```