```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

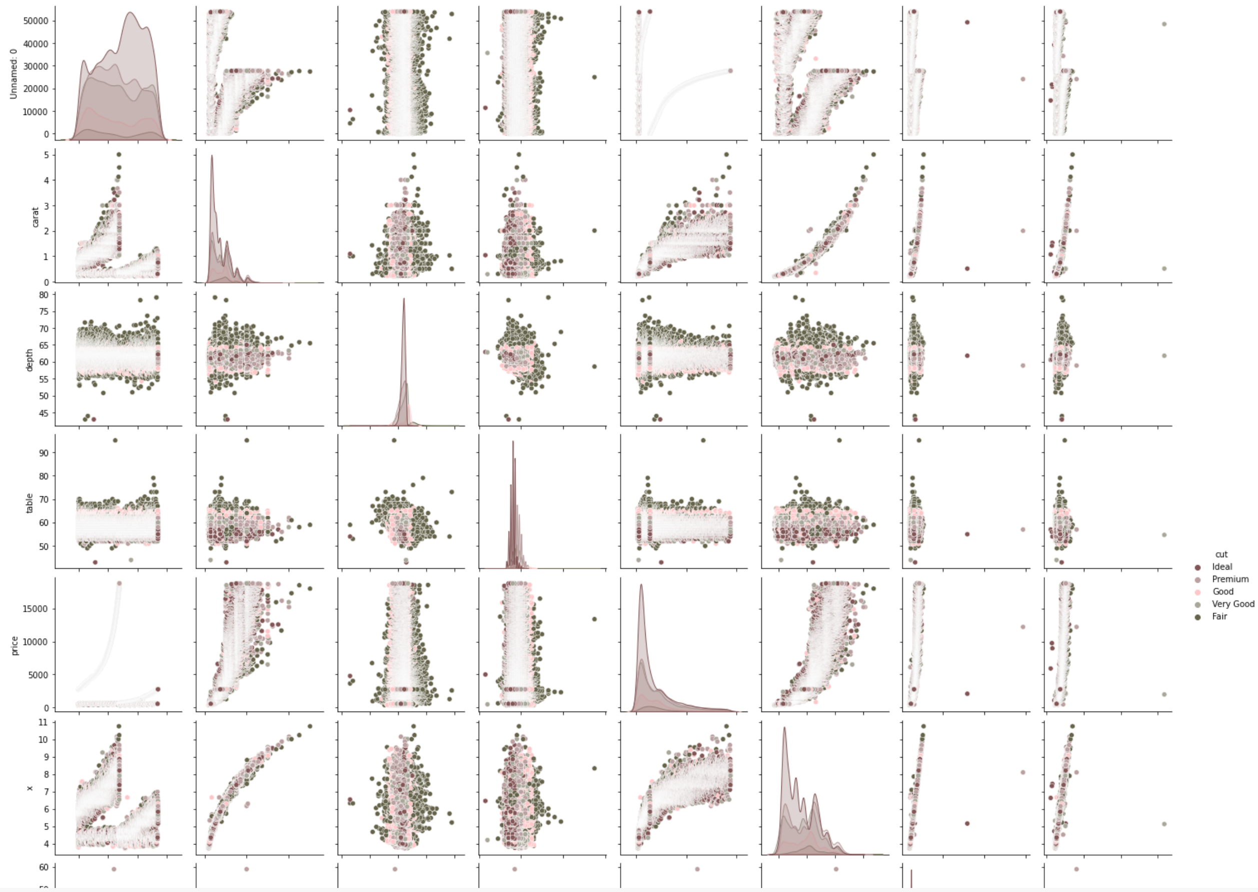
```
import numpy as np # linear algebra
import pandas as pd # data processing

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics

data = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/diamonds.csv")
data.head()
```

| | Unnamed: 0 | carat | cut | color | clarity | depth | table | price | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0.23 | Ideal | E | SI2 | 61.5 | 55.0 | 326 | 3.95 | 3.98 | 2.43 |
| **1** | 2 | 0.21 | Premium | E | SI1 | 59.8 | 61.0 | 326 | 3.89 | 3.84 | 2.31 |
| **2** | 3 | 0.23 | Good | E | VS1 | 56.9 | 65.0 | 327 | 4.05 | 4.07 | 2.31 |
| **3** | 4 | 0.29 | Premium | I | VS2 | 62.4 | 58.0 | 334 | 4.20 | 4.23 | 2.63 |
| **4** | 5 | 0.31 | Good | J | SI2 | 63.3 | 58.0 | 335 | 4.34 | 4.35 | 2.75 |

```
#Dropping dimentionless diamonds
data = data.drop(data[data["x"]==0].index)
data = data.drop(data[data["y"]==0].index)
data = data.drop(data[data["z"]==0].index)
```

```
shade = ["#835656", "#baa0a0", "#ffc7c8", "#a9a799", "#65634a"]#shades for hue
ax = sns.pairplot(data, hue= "cut",palette=shade)
```

```
ax = sns.pairplot(data, hue='cut',palette=shade)
```




```
#Dropping the outliers.
data = data[(data["depth"]<75)&(data["depth"]>45)]
data = data[(data["table"]<80)&(data["table"]>40)]
data = data[(data["x"]<30)]
data = data[(data["y"]<30)]
data = data[(data["z"]<30)&(data["z"]>2)]
```

Unnamed: 0 carat depth table price x y z

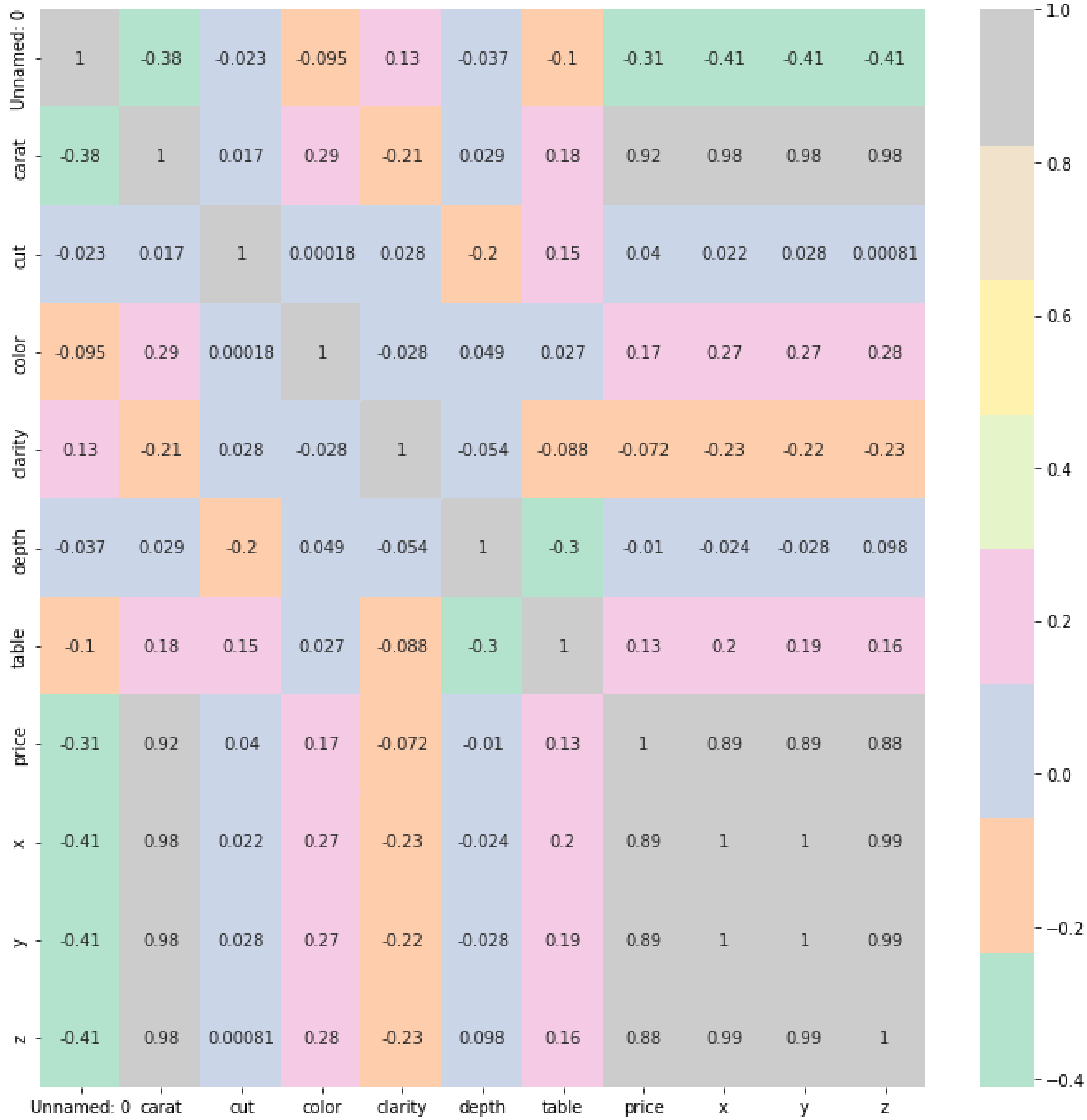
```
s = (data.dtypes =="object")
object_cols = list(s[s].index)
print("Categorical variables:")
print(object_cols)
```

```
Categorical variables:
['cut', 'color', 'clarity']
```

```
label_data = data.copy()

# Apply label encoder to each column with categorical data
label_encoder = LabelEncoder()
for col in object_cols:
    label_data[col] = label_encoder.fit_transform(label_data[col])
```

```
corrmat= label_data.corr()
f, ax = plt.subplots(figsize=(12,12))
sns.heatmap(corrmat,cmap="Pastel2",annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb9dab5ea10>
```




```
# Assigning the featurs as X and trarget as y
X= label_data.drop(["price"],axis =1)
y= label_data["price"]
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.25, random_state=7)

# Building pipelins of standard scaler and model for varios regressors.

pipeline_lr=Pipeline([("scalar1",StandardScaler()),
                     ("lr_classifier",LinearRegression())])

pipeline_dt=Pipeline([("scalar2",StandardScaler()),
                     ("dt_classifier",DecisionTreeRegressor())])
```

```
pipeline_rf=Pipeline([("scalar3",StandardScaler()),
                     ("rf_classifier",RandomForestRegressor())])


pipeline_kn=Pipeline([("scalar4",StandardScaler()),
                     ("rf_classifier",KNeighborsRegressor())])


pipeline_xgb=Pipeline([("scalar5",StandardScaler()),
                     ("rf_classifier",XGBRegressor())])

# List of all the pipelines
pipelines = [pipeline_lr, pipeline_dt, pipeline_rf, pipeline_kn, pipeline_xgb]

# Dictionary of pipelines and model types for ease of reference
pipe_dict = {0: "LinearRegression", 1: "DecisionTree", 2: "RandomForest",3: "KNeighbors", 4:

# Fit the pipelines
for pipe in pipelines:
    pipe.fit(X_train, y_train)

cv_results_rms = []
for i, model in enumerate(pipelines):
    cv_score = cross_val_score(model, X_train,y_train,scoring="neg_root_mean_squared_error",
    cv_results_rms.append(cv_score)
    print("%s: %f " % (pipe_dict[i], cv_score.mean()))
```

```
[05:49:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now d
LinearRegression: -1344.798387
DecisionTree: -51.813443
RandomForest: -36.084552
KNeighbors: -666.216132
[05:53:01] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now d
[05:53:03] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now d
[05:53:05] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now d
[05:53:07] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now d
[05:53:09] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now d
[05:53:11] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now d
[05:53:12] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now d
[05:53:14] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now d
[05:53:16] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now d
[05:53:18] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now d
XGBRegressor: -205.509411
```

```
pred = pipeline_xgb.predict(X_test)
print("R^2:",metrics.r2_score(y_test, pred))
print("Adjusted R^2:",1 - (1-metrics.r2_score(y_test, pred))*(len(y_test)-1)/(len(y_test)-X_t
print("MAE:",metrics.mean_absolute_error(y_test, pred))
print("MSE:",metrics.mean_squared_error(y_test, pred))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
R^2: 0.9973517661490602
Adjusted R^2: 0.997349799541418
MAE: 126.68033614855293
MSE: 41545.027262746946
RMSE: 203.8259729836876
```

0s completed at 12:02