Project - SHAI Internship 2024

Diamond price prediction 2024

G2

Frame the Problem & Select a Performance Measure

الهدف هو بناء نموذج يتوقع أسعار قطع الألماس، حيث يتم تطبيق النموذج على مجموعة بيانات خاصة بالاختبار ثم مقارنتها مع الأسعار الفعلية لقطع الألماس بوساطة استخدام مقياس الأداء جذر متوسط مربع الخطأ "RMSE" والذي يتم استخدامه بشكل عام ضمن مسائل الانحدار "Regression".

Get the Data

لدينا مجموعة بيانات تتضمن أسعار وسمات إضافية لما يقرب من 54000 قطعة ألماس، مقسمة إلى مجموعتين، الأولى يمكن من خلالها تطوير النموذج، والثانية مخصصة لتقييم دقة النموذج وارسال النتائج النهائية.

Take a Quick Look at the Data Structure

نظرة سريعة على البيانات:

```
train.head(3)
```

	Id	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	1.06	Ideal	I	SI2	61.8	57.0	4270	6.57	6.60	4.07
1	2	1.51	Premium	G	VVS2	60.9	58.0	15164	7.38	7.42	4.51
2	3	0.32	Ideal	F	VS2	61.3	56.0	828	4.43	4.41	2.71

وصف سريع للبيانات، يتضمن إجمالي عدد الصفوف وكل نوع من أنواع السمات وعدد القيم غير الخالية:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43152 entries, 0 to 43151
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Id       43152 non-null  int64  
 1   carat    43152 non-null  float64
 2   cut      43152 non-null  object 
 3   color    43152 non-null  object 
 4   clarity  43152 non-null  object 
 5   depth    43152 non-null  float64
 6   table    43152 non-null  float64
 7   price    43152 non-null  int64  
 8   x        43152 non-null  float64
 9   y        43152 non-null  float64
 10  z        43152 non-null  float64
dtypes: float64(6), int64(2), object(3)
memory usage: 3.6+ MB
```

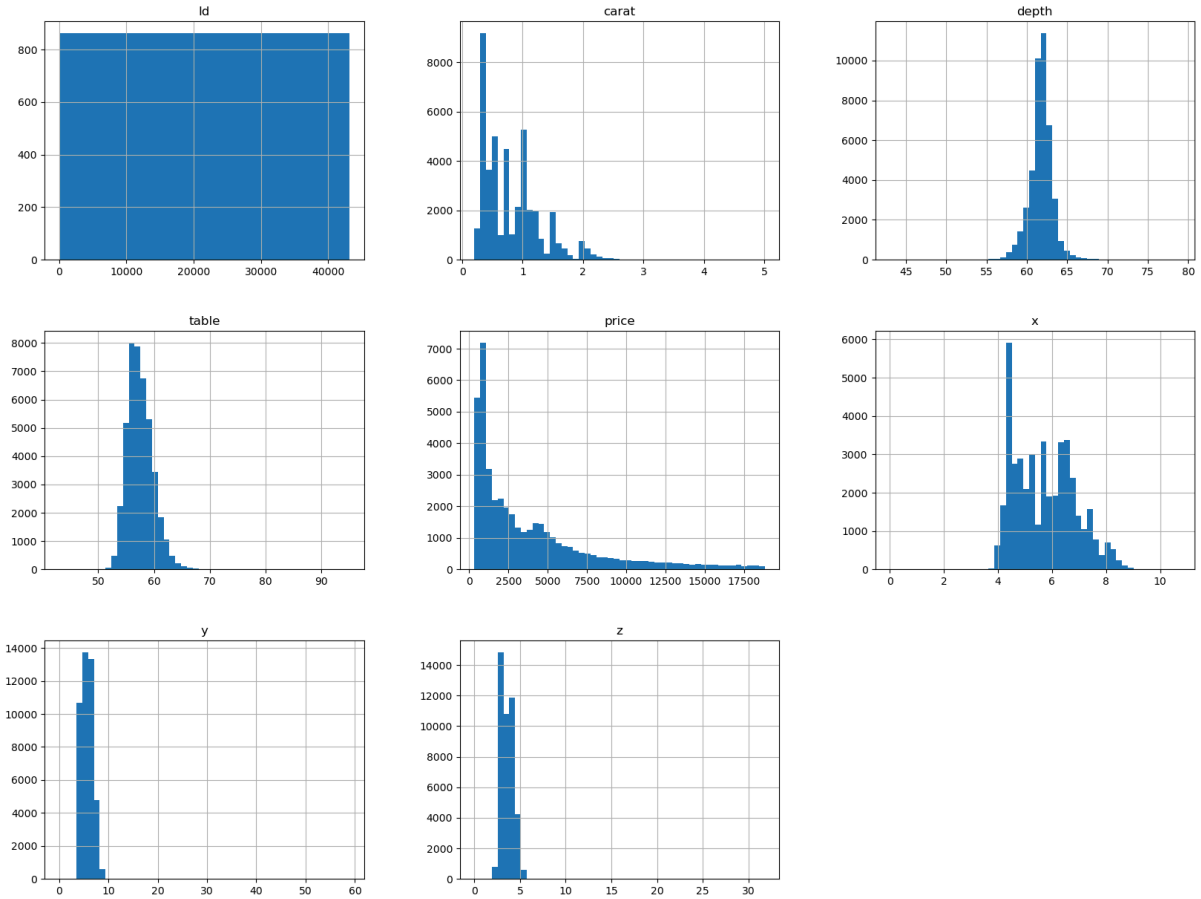
نلاحظ أن لا يوجد قيّم خالية وأنواع السمات مناسبة من أجل البدء بمرحلة استكشاف البيانات ولسنا بحاجة لتغيير النوع في أي من السمات.

ملخص للسمات العددية (numerical attributes):

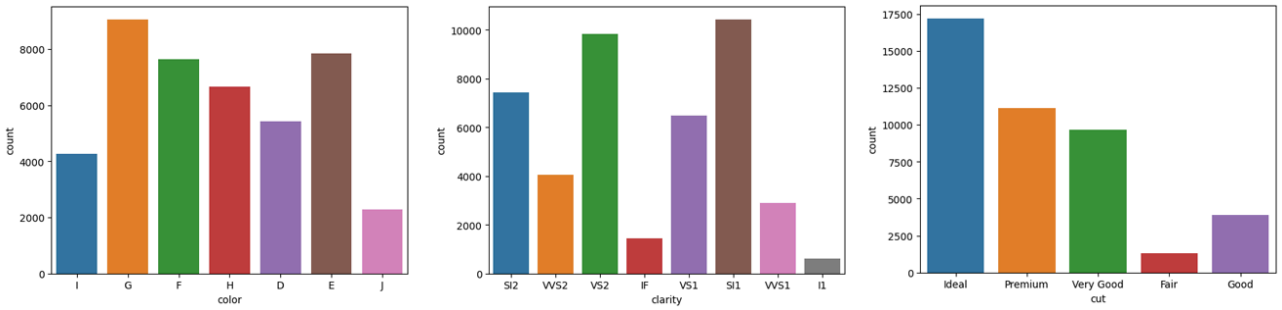
```
train.describe()
```

	Id	carat	depth	table	price	x	y	z
count	43152.000000	43152.000000	43152.000000	43152.000000	43152.000000	43152.000000	43152.000000	43152.000000
mean	21576.500000	0.797855	61.747177	57.458347	3929.491912	5.731568	5.735018	3.538568
std	12457.053745	0.473594	1.435454	2.233904	3985.527795	1.121279	1.148809	0.708238
min	1.000000	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	10788.750000	0.400000	61.000000	56.000000	947.750000	4.710000	4.720000	2.910000
50%	21576.500000	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	32364.250000	1.040000	62.500000	59.000000	5312.000000	6.540000	6.540000	4.040000
max	43152.000000	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

نلاحظ أن متوسط سعر قطع الألماس هو 3923.5، في مجال يتراوح بين 326 لأقل سعر وحتى 18823 لأغلى سعر لدينا، مع العلم أن 75% من قطع الألماس سعرها أقل من 5312، ويمكن رؤية هذه القيّم من خلال المدرجات التكرارية (Histogram).



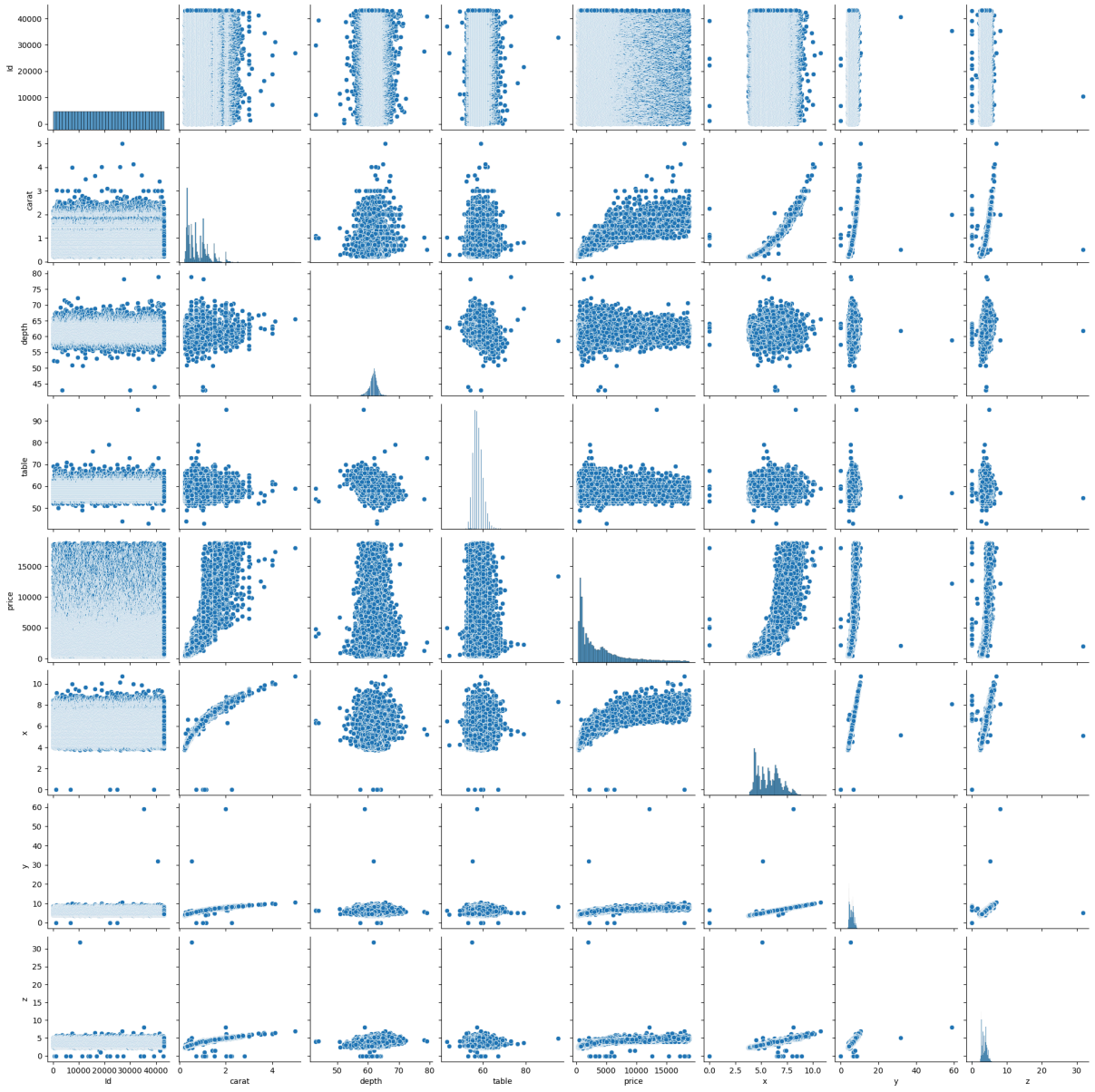
مخطط يوضح توزع البيانات الخاصة بالسمات الغير العددية (السمات الإسمية):



Looking for Correlations

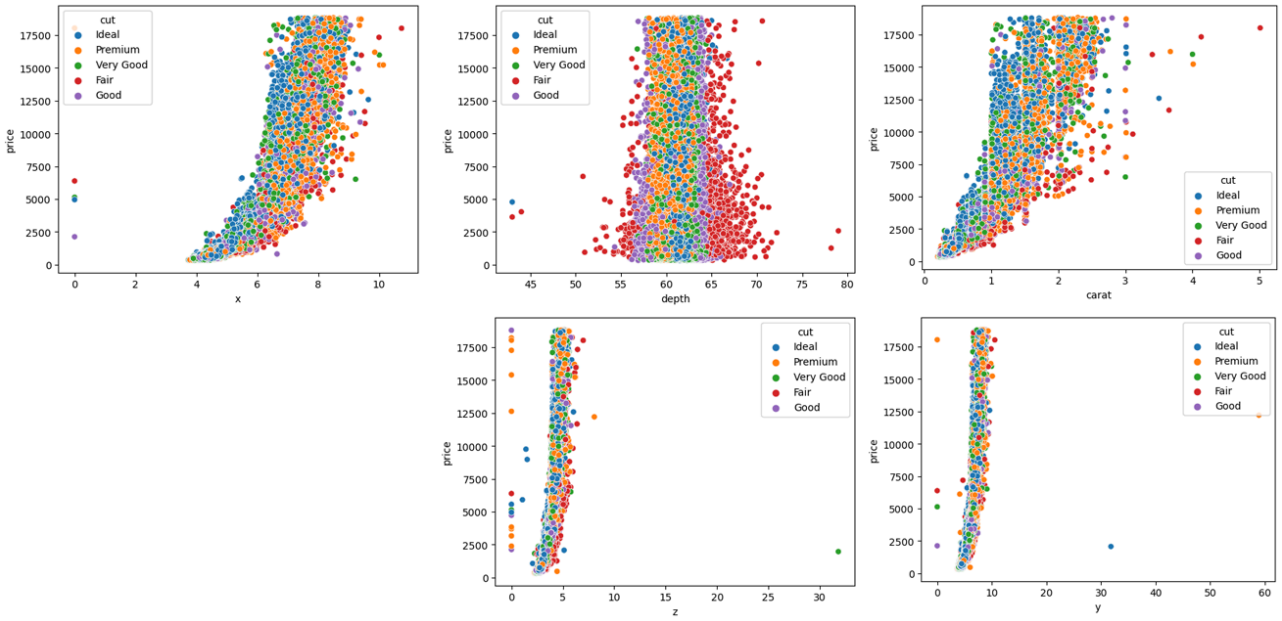
بعد الانتهاء من مرحلة استكشاف وفهم البيانات، وبما مجموعة البيانات لدينا ليست كبيرة جداً وعقدة، يمكن إيجاد الارتباط بين سمة "السعر" وباقي السمات الأخرى في مجموعة البيانات.

تم استخدام الدالة pairplot من مكتبة seaborn من أجل الحصول على عرض شامل للارتباط بين جميع السمات العددية.



نلاحظ وجود ارتباط إيجابي (علاقة طردية) بين سمة "price" من جهة، وبين سمات "carat"، و"x" من جهة أخرى.

ننتقل إلى السمات الاسمية (غير العددية) لإيجاد ارتباط بينها وبين سمة "السعر" وباقي السمات العددية الأخرى، مثلاً السمة "cut" وعلاقتها مع "السعر" والسمات الأخرى.

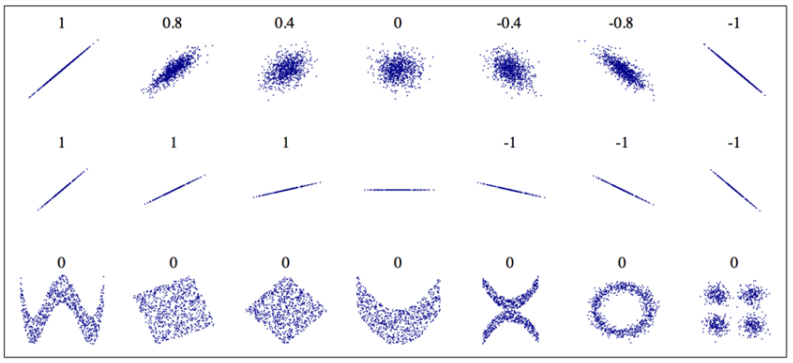


نلاحظ أن الارتباط بين السمات العددية والاسمية موجود تقريباً، لكنه غير قوي.

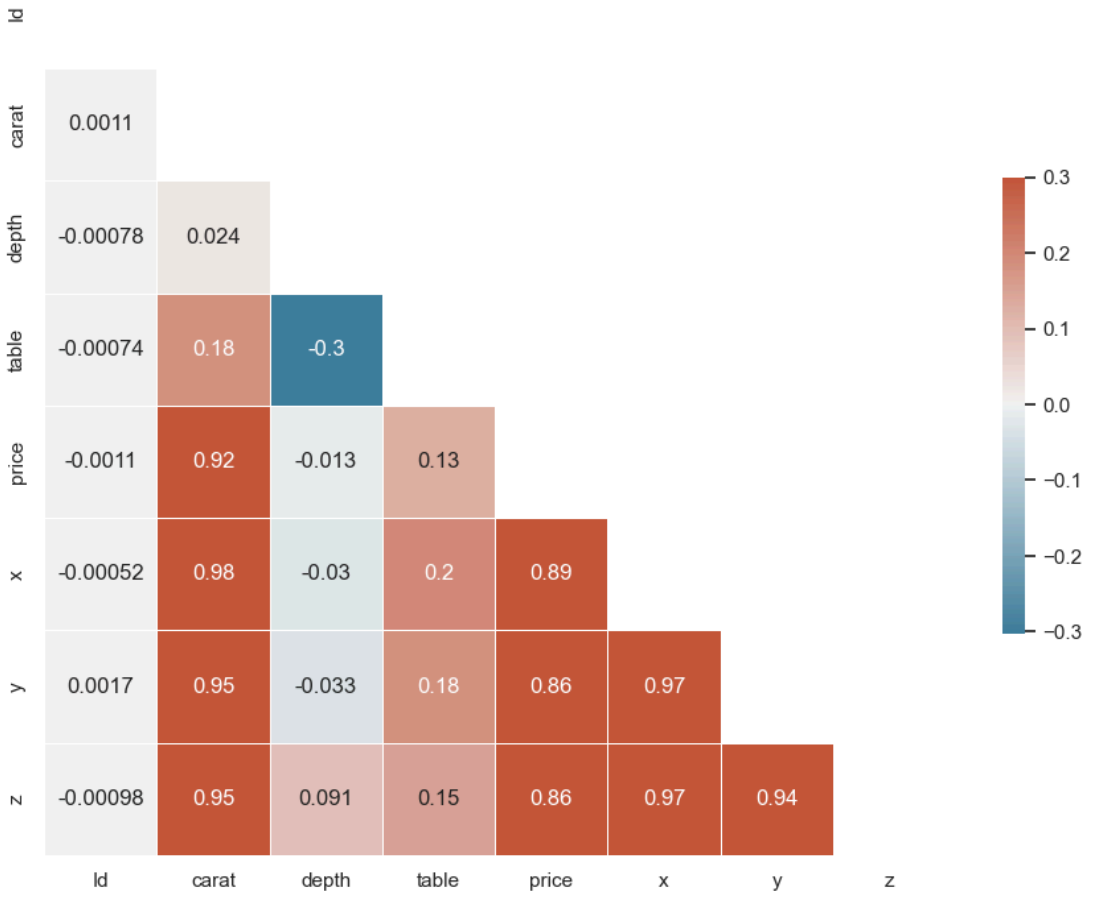
معامل الارتباط لسمة "السعر" مع السمات الأخرى:

```
corr_matrix = train.corr()
corr_matrix["price"].sort_values(ascending=False)
```

```
price     1.000000
carat     0.921911
x         0.885181
y         0.861354
z         0.857665
table     0.128501
Id       -0.001111
depth    -0.013137
```



مصفوفة الارتباطات لسمات العددية: حيث نلاحظ وجود ارتباط قوي بين "price" و "carat"، و"x"، و"y"، و"z".



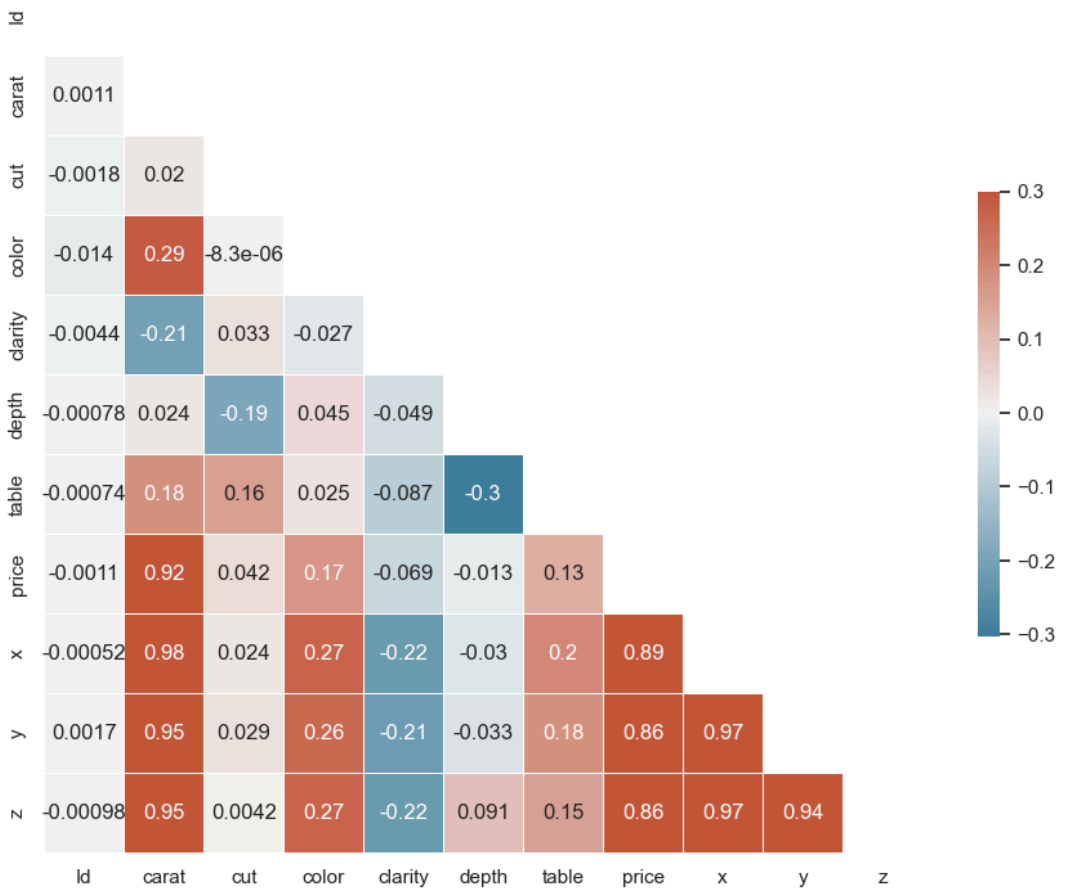
Data Cleaning

في هذه المرحلة يجب التخلص من القيم الفارغة والقيم المتطرفة ومعالجة القيم الإسمية وضبط "Feature Scaling"، لكي تصبح مناسبة من أجل عملية التدريب.

يتم معالجة القيم الاسمية عن طريق "LabelEncoder"، حيث يتم مقابلة كل تصنيف للبيانات برقم معين، وتصبح مجموعة البيانات بعد المعالجة:

	Id	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	1.06	2	5	3	61.8	57.0	4270	6.57	6.60	4.07
1	2	1.51	3	3	7	60.9	58.0	15164	7.38	7.42	4.51
2	3	0.32	2	2	5	61.3	56.0	828	4.43	4.41	2.71
3	4	0.53	2	3	5	61.2	56.0	1577	5.19	5.22	3.19
4	5	0.70	3	4	7	61.0	57.0	2596	5.76	5.72	3.50

مصفوفة الارتباطات بعد عملية تنظيف وتحضير البيانات:



Select and Train a Model

Create a Test Set and Train Set

تم اختيار السمات التي لها ارتباط طردي مع سمة "السعر" في البداية:

```
train_set, test_set = train_test_split(train, test_size=0.2,
random_state=42)

X_train = train_set.drop(['price','Id','clarity','depth'], axis=1)
y_train = train_set['price']
X_test = test_set.drop(['price','Id','clarity','depth'], axis=1)
y_test = test_set['price']
```

```
X_train.head()
```

	carat	cut	color	table	x	y	z
21805	1.588184	3	6	0.690124	1.487990	1.405805	1.526952
22939	0.511299	3	3	0.242472	0.649651	0.587557	0.651529
33888	0.891376	3	3	0.242472	1.050984	0.961862	1.004522
35779	-0.607817	2	4	-0.652832	-0.563263	-0.526654	-0.478048
20589	-0.818971	2	1	-1.548136	-0.821900	-0.848730	-0.802801

بعد أن أصبحت البيانات جاهزة، يمكن الآن اختيار نموذج تعلم آلي والمقارنة بينها من حيث RSME.

يمكن استخدام المكتبة **pycaret** والتي تساعد في المقارنة بين عدة نماذج بعد تدريبها على مجموعة البيانات الخاصة بالتدريب.

هنا نجد أفضل 15 نموذجاً يمكن تطبيقه على البيانات لدينا:

```
[ExtraTreesRegressor(n_jobs=-1, random_state=55),
 LGBMRegressor(n_jobs=-1, random_state=55),
 RandomForestRegressor(n_jobs=-1, random_state=55),
 GradientBoostingRegressor(random_state=55),
 DecisionTreeRegressor(random_state=55),
 AdaBoostRegressor(random_state=55),
 Lasso(random_state=55),
 LassoLars(random_state=55),
 Ridge(random_state=55),
 BayesianRidge(),
 LinearRegression(n_jobs=-1),
 Lars(random_state=55),
 ElasticNet(random_state=55),
 HuberRegressor(),
 KNeighborsRegressor(n_jobs=-1)]
```

- ## ExtraTreesRegressor

```
reg = ExtraTreesRegressor(n_estimators=100, random_state=55)
reg.fit(X_train, y_train)
```

```
Train RMSE = 49.11538516361631
Test RMSE = 1160.9000453836734
```

- ## LGBMRegressor

```
model = LGBMRegressor(metric='rmse')
model.fit(X_train, y_train)
```

```
Train RMSE = 66.52452744573787
Test RMSE = 1130.7274490799578
```

- ## RandomForestRegressor

```
regr = RandomForestRegressor(max_depth=10, random_state=0)
regr.fit(X_train, y_train)
```

```
Train RMSE = 150.24807202583114
Test RMSE = 1141.7564537758703
```

- ## RandomForestRegressor

```
regu = GradientBoostingRegressor(random_state=55)
regu.fit(X_train, y_train)
```

```
Train RMSE = 228.77819657720605
Test RMSE = 1147.0523968220978
```

- ## DecisionTreeRegressor

```
regressor = DecisionTreeRegressor(random_state=55)
regressor.fit(X_train, y_train)
```

```
Train RMSE = 4.002791678407103e-13
Test RMSE = 1137.5381769112144
```

بعد تجربة أغلب النماذج، تم تجربة نموذج تدريب إضافي وهو "XGBRegressor":

- **XGBRegressor**

```
cross_scores = cross_val_score(XGBRegressor(),X_train,y_train,cv=cvv)
```

Fine tune hyperparamters:

```
cvv=KFold(n_splits=6,shuffle=True,random_state=21)

param = {'learning_rate': [0.01, 0.1, 0.2],'max_depth': [3, 4, 5,7],
'n_estimators': [30, 50, 100]}

grid=GridSearchCV(XGBRegressor(),param,cv=cvv)
```

```
Train RMSE = 81.23453649972191
Test RMSE = 1130.8142767949785
```

تم إضافة السمات 'clarity' و'depth' مجدداً لمجموعة التدريب، ونلاحظ أن النتيجة أصبحت أفضل بعد تنفيذ ذات الـ code السابق.

```
Train RMSE = 413.0295749427162
Test RMSE = 528.9905435926516
```

حفظ توقعات النموذج ضمن ملف CSV:

```
submit_CSV = pd.DataFrame()
price = grid.predict(test.drop('Id',axis= 1))
submit_CSV['Id']= test['Id']
submit_CSV['price']= price
submit_CSV.to_csv('Sub4.csv',index= None)
```