Data Collection and Preprocessing Phase

Date	15 October 2024
Team ID	740663
Project Title	Predicting Diamond Prices With ANN Using Deep Learning.
Maximum Marks	6 Marks

Preprocessing Template

To preprocess data for predicting diamond prices, start by loading and exploring the dataset to identify missing values and outliers. Handle missing data by dropping or imputing them. Convert categorical variables like cut, colour, and clarity into numeric values using label encoding or one-hot encoding. Scale numeric features like carat, depth, and table using standardization to ensure uniformity. Address outliers with techniques like IQR filtering. Finally, split the data into training and testing sets for model development and evaluation.

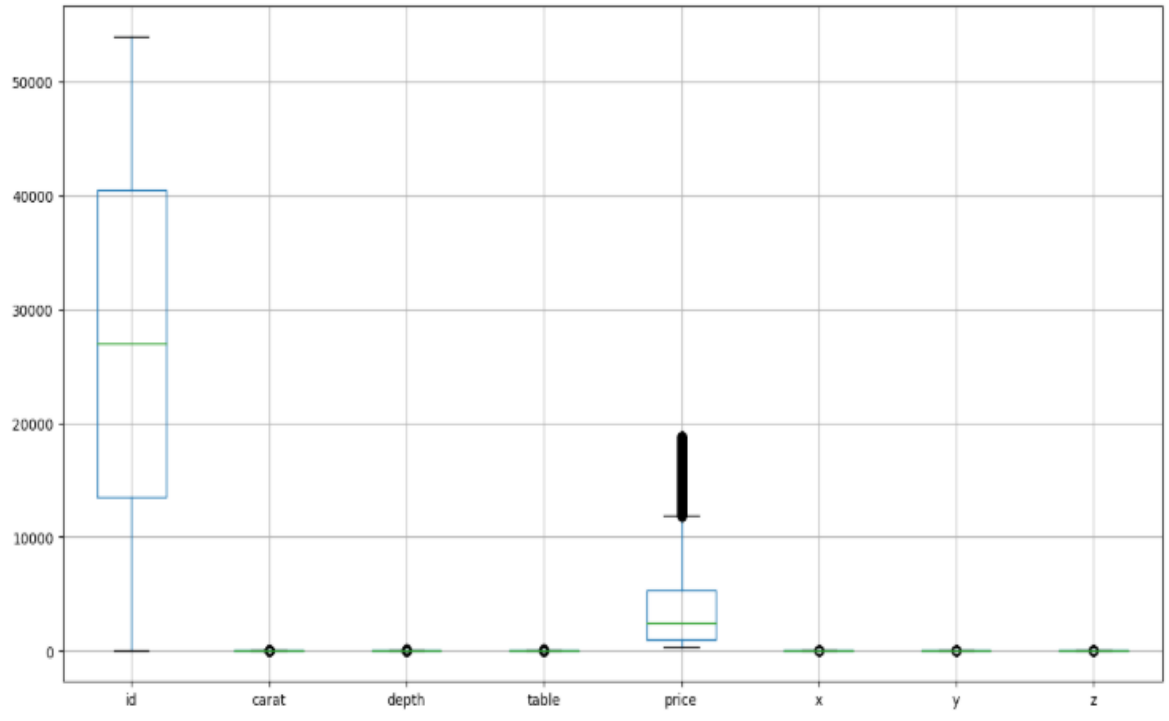
Section	Description
Data Overview	`[ ] data.describe()`

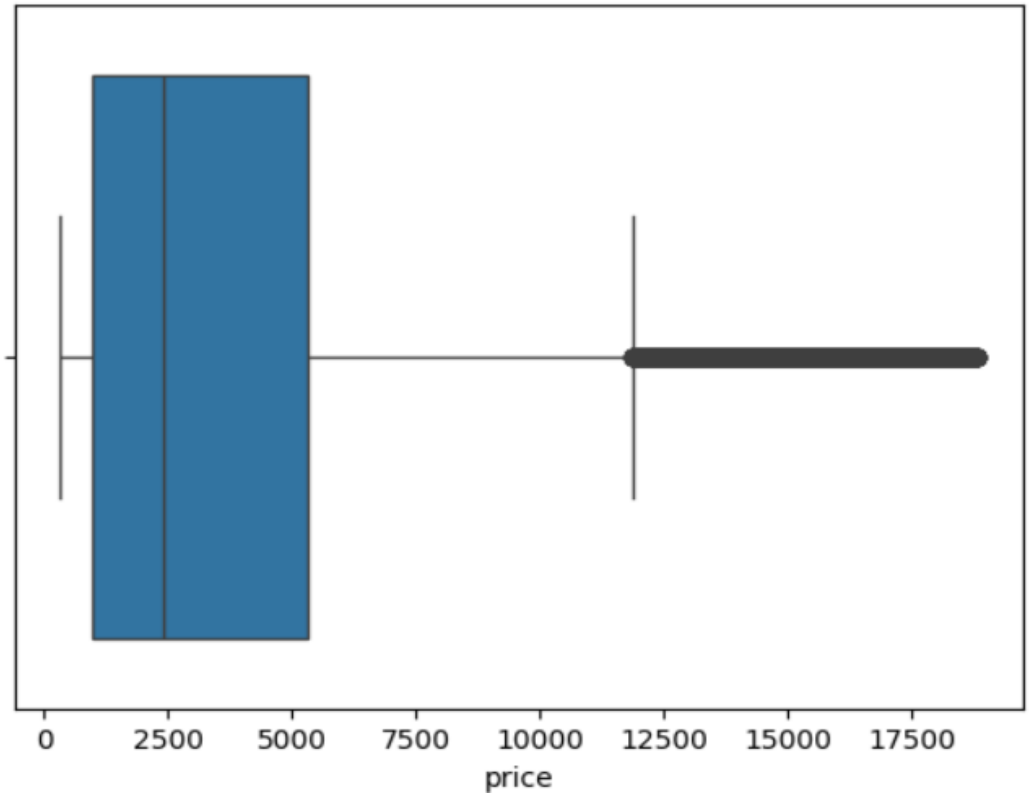
	id	carat	depth	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	26970.500000	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
std	15571.281097	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	1.000000	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	13485.750000	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	26970.500000	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	40455.250000	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	53940.000000	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

Boxplot

```
plt.figure(figsize=(15,8))s
data.boxplot()
```

```
<Axes: >
```

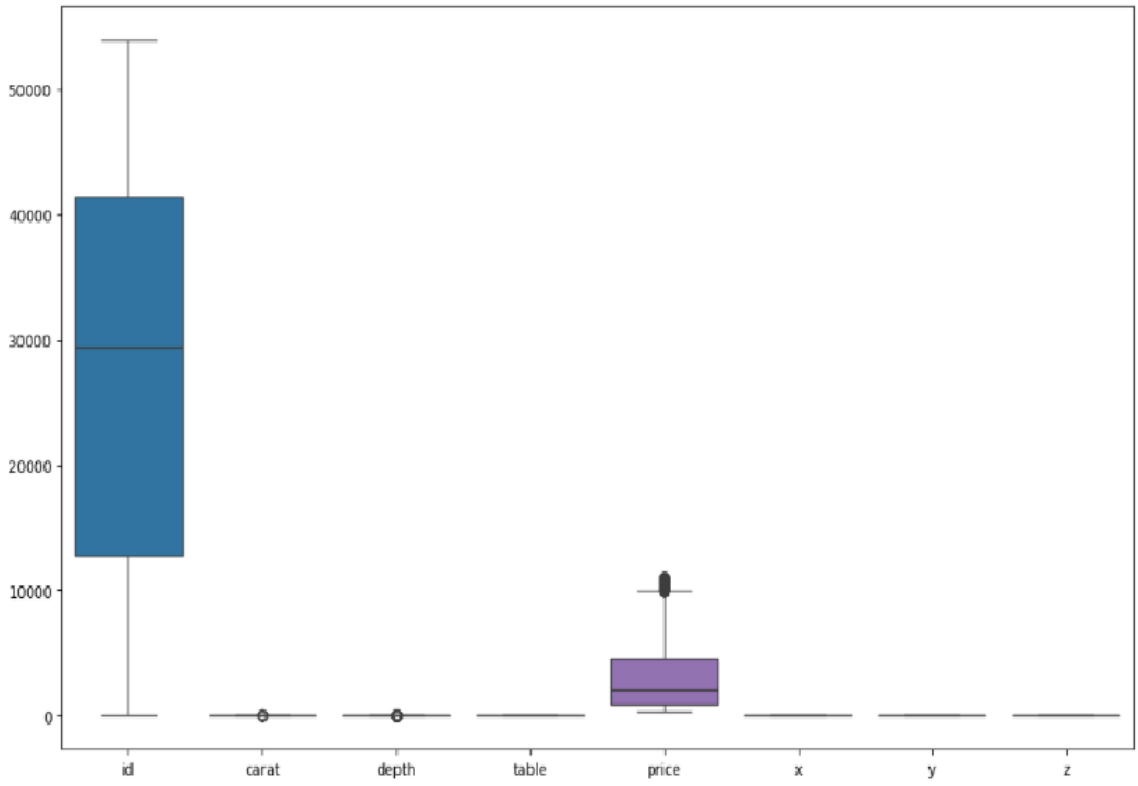


	[ ] `sns.boxplot(x="price",data=data)` <Axes: xlabel='price'> 

Outliers	`# Removing outliers from the specified numerical columns`
	`df_clean = remove_outliers(data, numerical_columns)`
	
	`# Display the cleaned data`
	`print(df_clean)`
	
	`plt.figure(figsize=(14, 8))`
	`sns.boxplot(data=df_clean)`
	`plt.show()`

	id	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
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
5	6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
...	...	...	...	...	...	...	...	...	...	...	...
53935	53936	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	53937	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	53938	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	53939	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	53940	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64

[46532 rows x 11 columns]

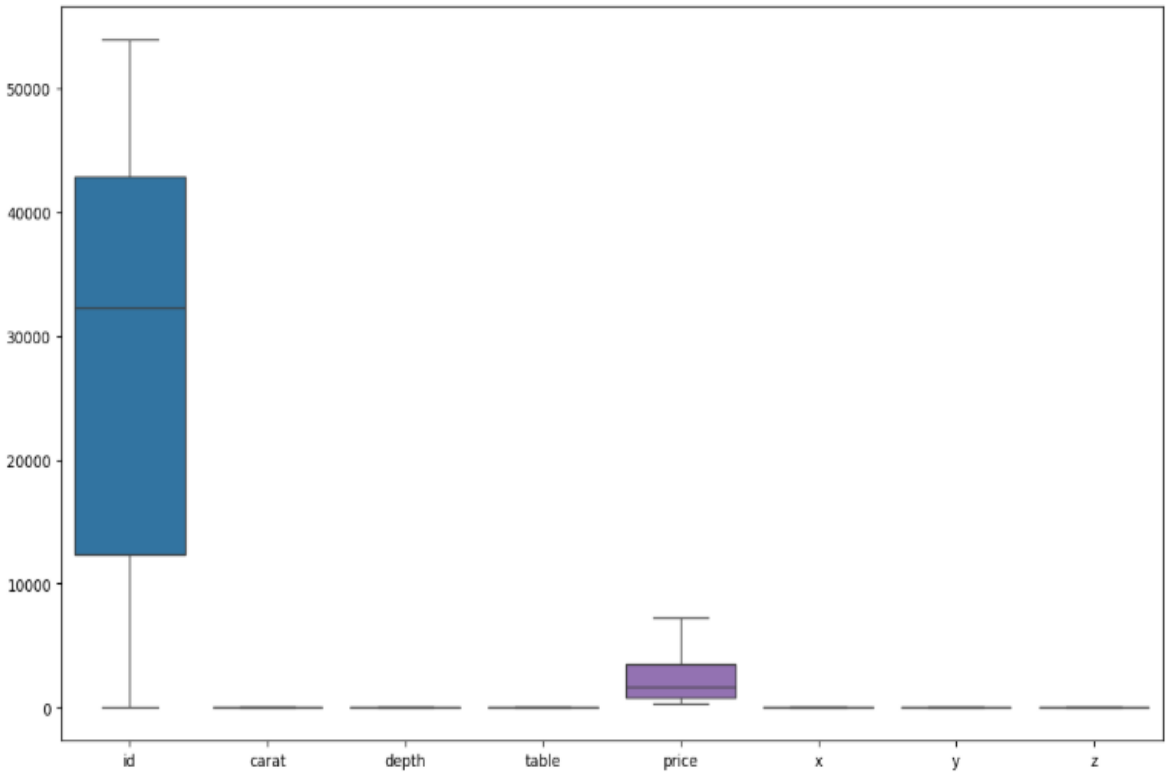


Mat Plot

```
          id  carat        cut color clarity  depth  table  price     x     y  \
0          1   0.23      Ideal     E     SI2   61.5   55.0    326  3.95  3.98
3          4   0.29    Premium     I     VS2   62.4   58.0    334  4.20  4.23
4          5   0.31       Good     J     SI2   63.3   58.0    335  4.34  4.35
5          6   0.24  Very Good     J    VVS2   62.8   57.0    336  3.94  3.96
6          7   0.24  Very Good     I    VVS1   62.3   57.0    336  3.95  3.98
...      ...    ...        ...   ...     ...    ...    ...    ...   ...   ...
53934  53935   0.72    Premium     D     SI1   62.7   59.0   2757  5.69  5.73
53935  53936   0.72      Ideal     D     SI1   60.8   57.0   2757  5.75  5.76
53936  53937   0.72       Good     D     SI1   63.1   55.0   2757  5.69  5.75
53938  53939   0.86    Premium     H     SI2   61.0   58.0   2757  6.15  6.12
53939  53940   0.75      Ideal     D     SI2   62.2   55.0   2757  5.83  5.87

          z
0      2.43
3      2.63
4      2.75
5      2.48
6      2.47
...     ...
53934  3.58
53935  3.50
53936  3.61
53938  3.74
53939  3.64

[29940 rows x 11 columns]
```



Data Preprocessing Code Screenshots

Loading Data

```
[ ] data=pd.read_csv('/content/diamonds.csv')
```

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

	id	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

```
[ ] data.tail()
```

	id	carat	cut	color	clarity	depth	table	price	x	y	z
53935	53936	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	53937	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	53938	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	53939	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	53940	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64

Checking Missing Values

```
[ ] data.isnull()
```

	id	carat	cut	color	clarity	depth	table	price	x	y	z
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
53935	False	False	False	False	False	False	False	False	False	False	False
53936	False	False	False	False	False	False	False	False	False	False	False
53937	False	False	False	False	False	False	False	False	False	False	False
53938	False	False	False	False	False	False	False	False	False	False	False
53939	False	False	False	False	False	False	False	False	False	False	False

53940 rows × 11 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   id       53940 non-null  int64
 1   carat    53940 non-null  float64
 2   cut      53940 non-null  object
 3   color    53940 non-null  object
 4   clarity  53940 non-null  object
 5   depth    53940 non-null  float64
 6   table    53940 non-null  float64
 7   price    53940 non-null  int64
 8   x        53940 non-null  float64
 9   y        53940 non-null  float64
 10  z        53940 non-null  float64
dtypes: float64(6), int64(2), object(3)
memory usage: 4.5+ MB
```

Data Preprocessing

```
[ ] x.head(10)
```

	carat	cut	color	clarity	depth	table	x	y	z
0	0.23	2	1	3	61.5	55.0	3.95	3.98	2.43
3	0.29	3	5	5	62.4	58.0	4.20	4.23	2.63
4	0.31	1	6	3	63.3	58.0	4.34	4.35	2.75
5	0.24	4	6	7	62.8	57.0	3.94	3.96	2.48
6	0.24	4	5	6	62.3	57.0	3.95	3.98	2.47
7	0.26	4	4	2	61.9	55.0	4.07	4.11	2.53
11	0.23	2	6	4	62.8	56.0	3.93	3.90	2.46
15	0.32	3	1	0	60.9	58.0	4.38	4.42	2.68
19	0.30	4	6	2	62.7	59.0	4.21	4.27	2.66
20	0.30	1	5	3	63.3	56.0	4.26	4.30	2.71

```
[ ]
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
x['cut']=le.fit_transform(x['cut'])
x['color']=le.fit_transform(x['color'])
x['clarity']=le.fit_transform(x['clarity'])
```

```
[ ] x
```

	carat	cut	color	clarity	depth	table	x	y	z
0	0.23	2	1	3	61.5	55.0	3.95	3.98	2.43
3	0.29	3	5	5	62.4	58.0	4.20	4.23	2.63
4	0.31	1	6	3	63.3	58.0	4.34	4.35	2.75
5	0.24	4	6	7	62.8	57.0	3.94	3.96	2.48
6	0.24	4	5	6	62.3	57.0	3.95	3.98	2.47
...	...	...	...	...	...	...	...	...	...
53934	0.72	3	0	2	62.7	59.0	5.69	5.73	3.58
53935	0.72	2	0	2	60.8	57.0	5.75	5.76	3.50
53936	0.72	1	0	2	63.1	55.0	5.69	5.75	3.61
53938	0.86	3	4	3	61.0	58.0	6.15	6.12	3.74
53939	0.75	2	0	3	62.2	55.0	5.83	5.87	3.64

29940 rows × 9 columns

Data Splitting

```
[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
    #why random state =42
```

```
[ ] from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    x_train = scaler.fit_transform(x_train)
    x_test = scaler.transform(x_test)
```

```
[ ] x
```

	carat	cut	color	clarity	depth	table	x	y	z
0	0.23	2	1	3	61.5	55.0	3.95	3.98	2.43
3	0.29	3	5	5	62.4	58.0	4.20	4.23	2.63
4	0.31	1	6	3	63.3	58.0	4.34	4.35	2.75
5	0.24	4	6	7	62.8	57.0	3.94	3.96	2.48
6	0.24	4	5	6	62.3	57.0	3.95	3.98	2.47
...	...	...	...	...	...	...	...	...	...
53934	0.72	3	0	2	62.7	59.0	5.69	5.73	3.58
53935	0.72	2	0	2	60.8	57.0	5.75	5.76	3.50
53936	0.72	1	0	2	63.1	55.0	5.69	5.75	3.61
53938	0.86	3	4	3	61.0	58.0	6.15	6.12	3.74
53939	0.75	2	0	3	62.2	55.0	5.83	5.87	3.64

Save Processed Data

```
[ ] import pickle
    import joblib
    joblib.dump(model,'model.joblib')
    joblib.dump(scaler, 'scaler.joblib')
    joblib.dump(le, 'label_encoder.joblib')
```

```
['label_encoder.joblib']
```