In [1]:
```
import pandas as pd
```

In [2]:
```
dataset = pd.read_csv("laptops_clean.CSV")
```

In [3]:
```
dataset.head()
```

Out[3]:

	name	screen diagonal	screen resolution	cpu	ram	storage type	storage size	price	
0	Lenovo IdeaPad 3 15IGL05 81WQ000JRK серый	15.6 дюйм	1366x768	Intel Celeron N4120	4 ГБ	HDD	1000 ГБ	149990	https://kaspi.kz/shc ideap
1	Lenovo IdeaPad L340-15API 81LW0068RK черный	15.6 дюйм	1920x1080	AMD Ryzen 5 3500U	8 ГБ	HDD	1000 ГБ	199990	https://kaspi.kz/shc ideap
2	Acer EX215-21 NX.EFUER.00J черный	15.6 дюйм	1920x1080	AMD A6 9220e	4 ГБ	SSD	256 ГБ	149990	https://kaspi.kz/s ex215-2
3	HP 250 G7 2M2Y9ES черный	15.6 дюйм	1280x720	Intel Pentium Silver N5030	8 ГБ	SSD	256 ГБ	179900	https://kaspi.kz 250-g7-2m
4	HP 250 G7 255J7ES черный	15.6 дюйм	1280x720	Intel Pentium Silver N5030	8 ГБ	SSD	128 ГБ	188800	https://kaspi.kz 250-g7-25

In [4]:
```
import matplotlib.pyplot as plt
```

In [5]:
```
import seaborn as sns
```

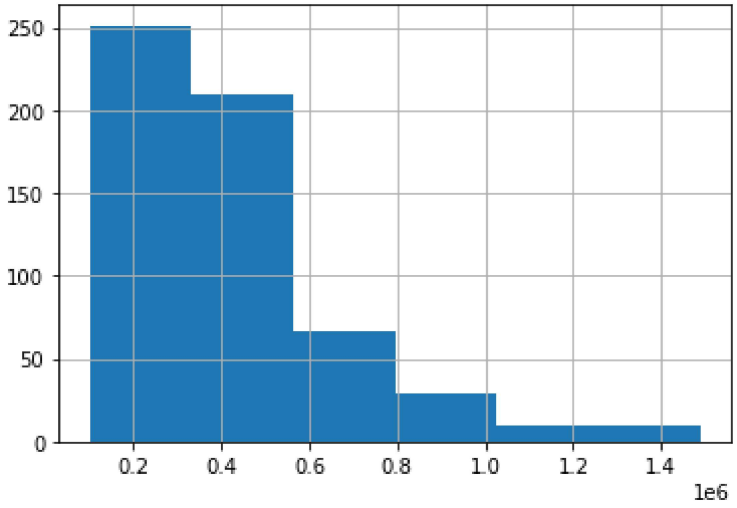
In [6]:
```
import pandas as pd
```

In [7]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

In [8]:
```
def score(y_true, y_pred):
    print("MSE:", mean_squared_error(y_true, y_pred))
    print("RMSE:", mean_squared_error(y_true, y_pred, squared=False))
    print("R2:", r2_score(y_true, y_pred))
```

In [10]:
```
data = pd.read_csv("laptops_clean.csv").drop("link", axis=1)
```

In [11]:
```
price_summary=data.describe()
price_summary
hist = data['price'].hist(bins=6)
plt.show()
```



In [12]:
```
# Not Implemented
data['name'][1:10]
data = data[~data['gpu'].str.contains("\+")]
data['name'] = data['name'].str.split(expand=True)[0]
data['name'][1:10]
data['gpu'][1:10]
data['gpu company'] = data['gpu'].str.split(expand=True)[0]
data['gpu company'][1:10]
data['cpu'][1:10]
data['cpu company'] = data['cpu'].str.split(expand=True)[0]
data['cpu company'][1:10]
type(data['price'])
data['price'] = data['price'].astype(float)
type(data['price'])
```

Out[12]:
```
pandas.core.series.Series
```

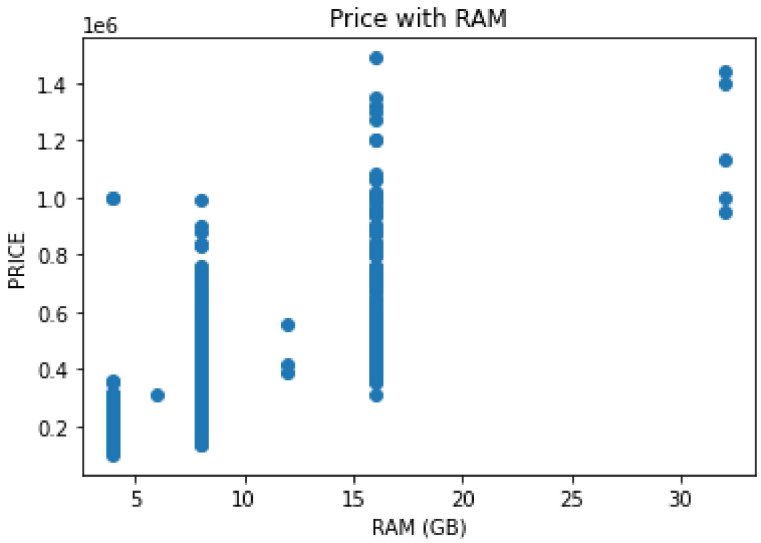
In [13]:
```
data['ram'][1:10]
data['ram'] = data['ram'].str.split(expand=True)[0]
data['ram'] = data['ram'].astype(float)
data['ram'][1:10]
```

Out[13]:
```
1    8.0
2    4.0
3    8.0
4    8.0
5    4.0
6    4.0
7    8.0
8    8.0
9    8.0
Name: ram, dtype: float64
```

In [28]:
```
data['storage size'][1:10]
data['storage size'] = data['storage size'].str.split(expand=True)[0]
data['storage size'] = data['storage size'].astype(float)
data['storage size'][1:10]
```

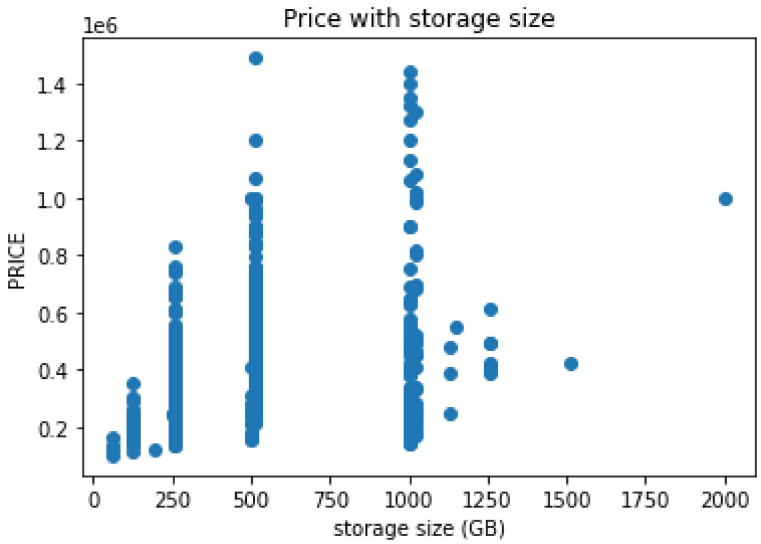
Out[28]:
```
1    1000.0
2     256.0
3     256.0
4     128.0
5    1000.0
6      64.0
7    1000.0
8     128.0
9     256.0
Name: storage size, dtype: float64
```

In [29]:
```
plt.scatter(data['ram'].to_frame(), data['price'].to_frame())
plt.title('Price with RAM')
plt.xlabel("RAM (GB)")
plt.ylabel("PRICE")
plt.show()
```



In [30]:
```
plt.scatter(data['storage size'].to_frame(), data['price'].to_frame())
plt.title('Price with storage size')
plt.xlabel("storage size (GB)")
```

```
plt.ylabel("PRICE")
plt.show()
datax=data.sort_values(by=['price'])
```



In [31]:
```
label_columns = ['screen diagonal', 'screen resolution', 'cpu', 'ram',
'storage size', 'gpu']
data[label_columns] =
data[label_columns].apply(LabelEncoder().fit_transform)
```

In [32]:
```
ohe_columns = ['storage type', 'gpu company', 'cpu company', 'name']
ohe_df = pd.get_dummies(data[ohe_columns])
data = pd.concat([data.drop(ohe_columns, axis=1), ohe_df], axis=1)
```

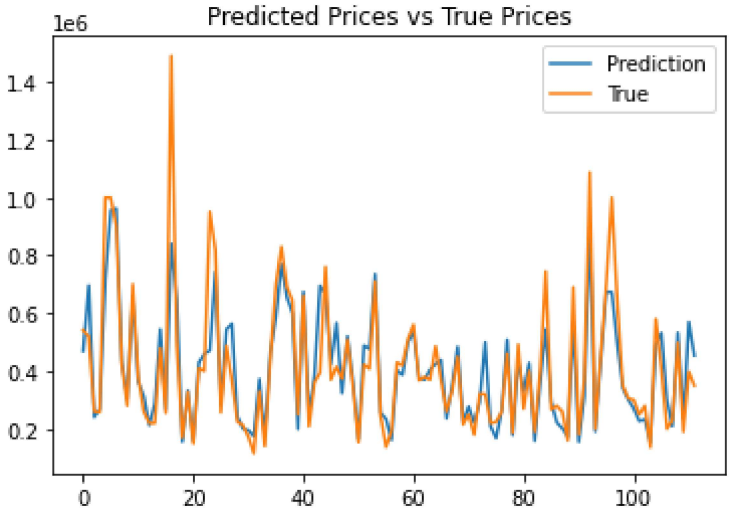
In [33]:
```
train_x, test_x, train_y, test_y = train_test_split(data.drop('price',
axis=1),
                                                    data['price'],
                                                    test_size=0.2,
                                                    random_state=0)
```

In [34]:
```
model = RandomForestRegressor(n_estimators=200,n_jobs=10,
random_state=0)
model.fit(train_x, train_y)
y_true=test_y
y_pred=model.predict(test_x)
print("R2:", r2_score(y_true, y_pred))
```

```
R2: 0.7777720556238212
```

In [35]:
```
y_true_n=y_true.to_numpy()
plt.plot(y_pred, label='Prediction')
plt.plot(y_true_n, label='True')
```

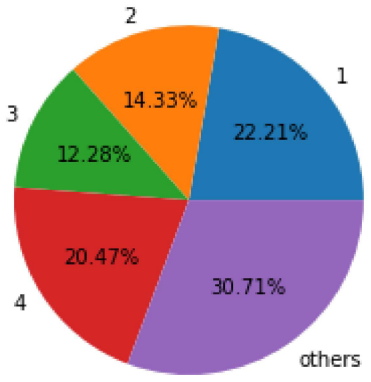
```
plt.title('Predicted Prices vs True Prices')
plt.legend()
plt.show()
```



In [38]:
```
import matplotlib.pyplot as plt
```

In [39]:
```
labels = ['1', '2', '3', '4', 'others']
share = [10.85, 7, 6, 10, 15]
plt.pie(x=share, labels=labels, autopct='%.2f%%')
```

Out[39]:
```
([<matplotlib.patches.Wedge at 0x1b6ccdd3280>,
  <matplotlib.patches.Wedge at 0x1b6ccde6670>,
  <matplotlib.patches.Wedge at 0x1b6ccde6d90>,
  <matplotlib.patches.Wedge at 0x1b6ccde74f0>,
  <matplotlib.patches.Wedge at 0x1b6ccde7c10>],
 [Text(0.8429014173459634, 0.7067653080310086, '1'),
  Text(-0.29862718676382566, 1.0586887188052603, '2'),
  Text(-0.985743914090406, 0.48816896238262264, '3'),
  Text(-0.9265937474080486, -0.5928102793173459, '4'),
  Text(0.6266615921074846, -0.9040438313363532, 'others')],
 [Text(0.4597644094614345, 0.3855083498350955, '22.21%'),
  Text(-0.16288755641663216, 0.5774665738937783, '14.33%'),
  Text(-0.5376784985947668, 0.2662739794814305, '12.28%'),
  Text(-0.505414771313481, -0.323351061445825, '20.47%'),
  Text(0.34181541387680975, -0.4931148170925563, '30.71%')])
```



In [ ]:

```
```