Laptop Price Data Project

June 25, 2024

1 Laptop Price Data Project

The aim of this study is to utilize mechine learning techniques to uncover and analyze the most relevant factors affecting the laptop price.

1.1 Part 1: Load the Data

This study is using the Laptop Price Dataset available from Kaggle, (URL: https://www.kaggle.com/datasets/gyanprakashkushwaha/laptop-price-prediction-cleaned-dataset).

[1]:
```
import scipy as sp
import scipy.stats as stats
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import copy
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

[2]:
```
df = pd.read_csv('data/laptop_data.csv')
df.head()
```

[2]:
```
  Company   TypeName  Ram  Weight      Price  TouchScreen  Ips         Ppi  \
0   Apple  Ultrabook    8    1.37  11.175755            0    1  226.983005
1   Apple  Ultrabook    8    1.34  10.776777            0    0  127.677940
2      HP   Notebook    8    1.86  10.329931            0    0  141.211998
3   Apple  Ultrabook   16    1.83  11.814476            0    1  220.534624
4   Apple  Ultrabook    8    1.37  11.473101            0    1  226.983005

       Cpu_brand  HDD  SSD Gpu_brand      Os
0  Intel Core i5    0  128     Intel     Mac
1  Intel Core i5    0    0     Intel     Mac
2  Intel Core i5    0  256     Intel  Others
```

```
3  Intel Core i7    0  512       AMD     Mac
4  Intel Core i5    0  256     Intel     Mac
```

1.2 Part 2: Check the Data Types

[3]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1273 entries, 0 to 1272
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Company      1273 non-null   object
 1   TypeName     1273 non-null   object
 2   Ram          1273 non-null   int64
 3   Weight       1273 non-null   float64
 4   Price        1273 non-null   float64
 5   TouchScreen  1273 non-null   int64
 6   Ips          1273 non-null   int64
 7   Ppi          1273 non-null   float64
 8   Cpu_brand    1273 non-null   object
 9   HDD          1273 non-null   int64
 10  SSD          1273 non-null   int64
 11  Gpu_brand    1273 non-null   object
 12  Os           1273 non-null   object
dtypes: float64(3), int64(5), object(5)
memory usage: 129.4+ KB
```

Based on the information of the dataset, there is no error found in data type.

1.3 Part 3: Data Cleaning

We can drop the features that are unnecessary. `Company`, `TypeName`, `Cpu_brand`, `Gpu_brand` and `Os` are object and not a meaningful feature so we can remove it and replace the df.

[4]:
```python
df = df.drop(columns=['Company', 'TypeName', 'Cpu_brand', 'Gpu_brand','Os'])
```

Reorder the columns so the `Price` (the target variable) is the first column.

[5]:
```python
df = df.iloc[:,[2,0,1,3,4,5,6,7]]
```

[6]:
```python
df.head()
```

[6]:
```
       Price  Ram  Weight  TouchScreen  Ips         Ppi  HDD  SSD
0  11.175755    8    1.37            0    1  226.983005    0  128
1  10.776777    8    1.34            0    0  127.677940    0    0
```

```
2  10.329931    8     1.86              0     0  141.211998    0  256
3  11.814476   16     1.83              0     1  220.534624    0  512
4  11.473101    8     1.37              0     1  226.983005    0  256
```

1.4 Part 4: Correlation Matrix

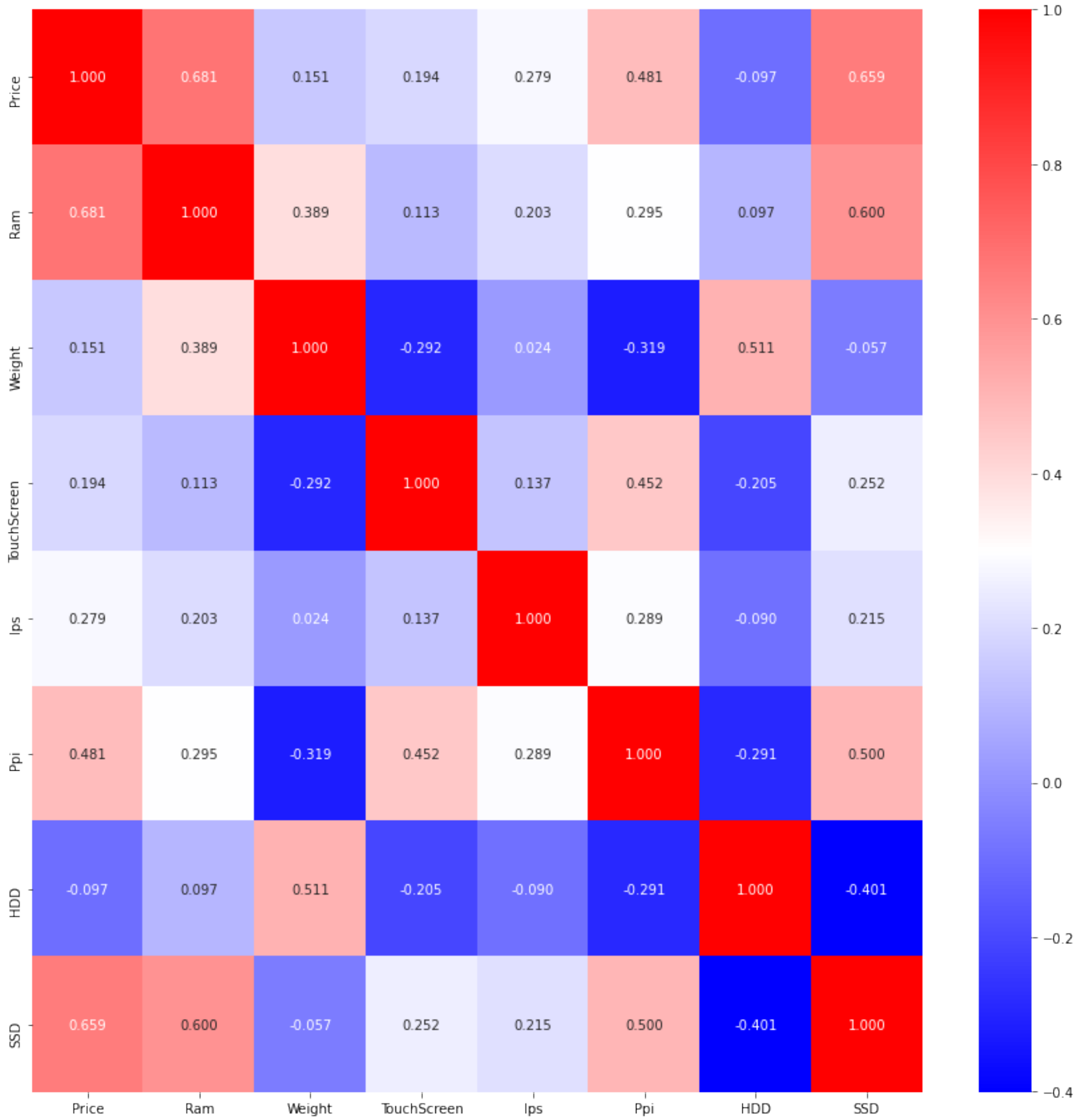
[7]:
```python
df.corr()
```

[7]:
```
                Price       Ram    Weight  TouchScreen       Ips       Ppi  \
Price        1.000000  0.680519  0.151386     0.194289  0.279240  0.480687
Ram          0.680519  1.000000  0.389134     0.113316  0.202809  0.294927
Weight       0.151386  0.389134  1.000000    -0.292288  0.023966 -0.319499
TouchScreen  0.194289  0.113316 -0.292288     1.000000  0.136973  0.452107
Ips          0.279240  0.202809  0.023966     0.136973  1.000000  0.288833
Ppi          0.480687  0.294927 -0.319499     0.452107  0.288833  1.000000
HDD         -0.097361  0.097340  0.510876    -0.205105 -0.090411 -0.290774
SSD          0.658808  0.599552 -0.056985     0.252142  0.215197  0.499899

                  HDD       SSD
Price       -0.097361  0.658808
Ram          0.097340  0.599552
Weight       0.510876 -0.056985
TouchScreen -0.205105  0.252142
Ips         -0.090411  0.215197
Ppi         -0.290774  0.499899
HDD          1.000000 -0.400625
SSD         -0.400625  1.000000
```

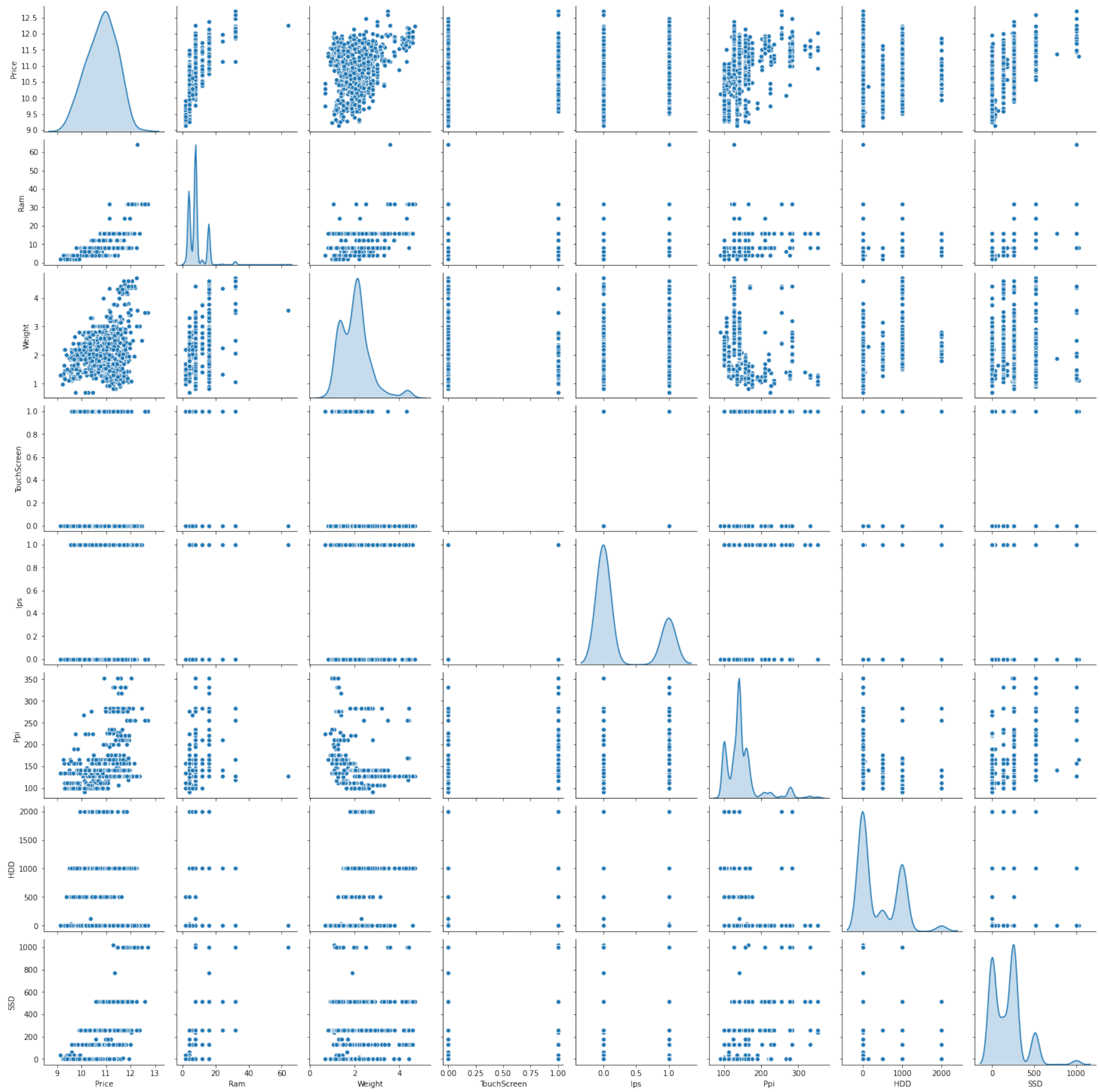
1.5 Part 5: Display the Correlation Matrix as Heat Map

[8]:
```python
fig, ax = plt.subplots(figsize=(15,15))
hm = sns.heatmap(df.corr(), fmt='.3f', cmap='bwr', annot=True, ax=ax,␣
 ↪xticklabels='auto', yticklabels='auto')
```



1.6 Part 5: Display the Correlation Matrix as Pair Plot

[9]:
```
selected_columns = df.columns[:10]
df_selected = df[selected_columns]
sns.pairplot(df_selected, diag_kind='kde')
plt.show()
```



Based on the heat map and pair plot, the best guess predictor is `Ram`, followed by `SSD` and `Ppi`.

1.7 Part 6: Simple linear regression

Firstly let's split the data to X_train and X_test.

[10]:
```
from sklearn.model_selection import train_test_split

X_train, X_test = train_test_split(df, test_size=0.2)
print("Length of X_train:", len(X_train))
print("Length of X_test:", len(X_test))
```

```
Length of X_train: 1018
Length of X_test: 255
```

[11]:
```
model = smf.ols(formula='Price ~ Ram', data=X_train).fit()

print(model.summary())

adj_R2 = model.rsquared_adj
print("Adjusted R-squared value:", adj_R2)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.459
Model:                            OLS   Adj. R-squared:                  0.459
Method:                 Least Squares   F-statistic:                     862.3
Date:                Mon, 24 Jun 2024   Prob (F-statistic):          9.99e-138
Time:                        12:11:51   Log-Likelihood:                -649.15
No. Observations:                1018   AIC:                             1302.
Df Residuals:                    1016   BIC:                             1312.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     10.1270      0.028    366.849      0.000      10.073      10.181
Ram            0.0819      0.003     29.364      0.000       0.076       0.087
==============================================================================
Omnibus:                       67.417   Durbin-Watson:                   1.931
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              133.658
Skew:                          -0.434   Prob(JB):                     9.47e-30
Kurtosis:                       4.548   Cond. No.                         19.2
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
Adjusted R-squared value: 0.4585445181447878
```

Then we create a Simple Linear Regression with formula `Price ~ SSD`.

[12]:
```
model = smf.ols(formula='Price ~ SSD', data=X_train).fit()

print(model.summary())

adj_R2 = model.rsquared_adj
print("Adjusted R-squared value:", adj_R2)
```

```
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:                  Price   R-squared:                       0.433
Model:                            OLS   Adj. R-squared:                  0.433
Method:                 Least Squares   F-statistic:                     777.2
Date:                Mon, 24 Jun 2024   Prob (F-statistic):          1.72e-127
Time:                        12:11:51   Log-Likelihood:                -672.74
No. Observations:                1018   AIC:                             1349.
Df Residuals:                    1016   BIC:                             1359.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     10.4167      0.021    505.483      0.000      10.376      10.457
SSD            0.0022   7.84e-05     27.879      0.000       0.002       0.002
==============================================================================
Omnibus:                        2.130   Durbin-Watson:                   1.966
Prob(Omnibus):                  0.345   Jarque-Bera (JB):                2.106
Skew:                          -0.070   Prob(JB):                        0.349
Kurtosis:                       2.826   Cond. No.                         369.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
Adjusted R-squared value: 0.432865330804283
```

Then we create a Simple Linear Regression with formula `Price ~ Ppi`.

[13]:
```python
model = smf.ols(formula='Price ~ Ppi', data=X_train).fit()

print(model.summary())

adj_R2 = model.rsquared_adj
print("Adjusted R-squared value:", adj_R2)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.221
Model:                            OLS   Adj. R-squared:                  0.220
Method:                 Least Squares   F-statistic:                     288.4
Date:                Mon, 24 Jun 2024   Prob (F-statistic):           4.03e-57
Time:                        12:11:51   Log-Likelihood:                -834.75
No. Observations:                1018   AIC:                             1674.
Df Residuals:                    1016   BIC:                             1683.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
```

```
------------------------------------------------------------------------------
Intercept      9.8130      0.062    159.008      0.000       9.692       9.934
Ppi            0.0069      0.000     16.981      0.000       0.006       0.008
==============================================================================
Omnibus:                        2.642   Durbin-Watson:                   2.043
Prob(Omnibus):                  0.267   Jarque-Bera (JB):                2.510
Skew:                          -0.115   Prob(JB):                        0.285
Kurtosis:                       3.081   Cond. No.                         545.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
Adjusted R-squared value: 0.22030696567962438
```

We can see that when we build the Simple Linear Regression Model with those 3 features that has highest corrlation with Price, the R-squared value was not large.

`Price ~ Ram` has highest R-squared value of 0.483.

`Price ~ SSD` has highest R-squared value of 0.437.

`Price ~ Ppi` has highest R-squared value of 0.215.

Therefore we can build a Multi-Linear Regression Model to see if it can improve the accuracy.

1.8 Part 7: Multi-Linear Regression Model

[14]:
```
# Initialize variables to store the best R² value and corresponding feature
best_r_squared = -1  # Start with the lowest possible value
best_predictor = ''

# Instantiate the regression model
model = LinearRegression()

# Loop through each column (except 'Price' as it's the target)
for column in df.columns:
    if column != 'Price' and df[column].dtype in ['float64', 'int64']:  #␣
↪Ensure the column is numeric
        X = df[[column]]  # Feature matrix
        y = df['Price']  # Target variable

        # Fit the model
        model.fit(X, y)

        # Predict the mpg
        y_pred = model.predict(X)

        # Calculate R² value
```

```
        r_squared = r2_score(y, y_pred)

        # Check if this R² is the best we've seen so far
        if r_squared > best_r_squared:
            best_r_squared = r_squared
            best_predictor = column

# Print the best predictor and its R² value
print(f'Best Predictor: {best_predictor}')
print(f'Best R² Value: {best_r_squared}')
```

```
Best Predictor: Ram
Best R² Value: 0.46310600112573364
```

[15]:
```
best_degree = 0
best_r_squared = 0

for degree in range(1, 21):
    X = np.column_stack([np.power(df[best_predictor], i) for i in range(1,␣
 ↪degree + 1)])
    y = df['Price']

    # Add a constant to the model (intercept)
    X = sm.add_constant(X)

    # Fit the model
    model = sm.OLS(y, X).fit()

    # Get the R² value
    r_squared = model.rsquared

    # Check if this R² is the best we've seen so far
    if r_squared > best_r_squared:
        best_r_squared = r_squared
        best_degree = degree

# Print the best degree and its R² value
print(f'Best Degree: {best_degree}')
print(f'Best R² Value: {best_r_squared}')
```

```
Best Degree: 8
Best R² Value: 0.6098242522793986
```

[16]:
```
model = smf.ols('Price ~ Ram + Weight + TouchScreen + Ips + Ppi + HDD + SSD',␣
 ↪data=df).fit()
```

```python
# Print the summary of the model
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.603
Model:                            OLS   Adj. R-squared:                  0.601
Method:                 Least Squares   F-statistic:                     274.8
Date:                Mon, 24 Jun 2024   Prob (F-statistic):          9.67e-249
Time:                        12:11:56   Log-Likelihood:                -607.93
No. Observations:                1273   AIC:                             1232.
Df Residuals:                    1265   BIC:                             1273.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept       9.5797      0.067    143.904      0.000       9.449       9.710
Ram             0.0454      0.003     13.536      0.000       0.039       0.052
Weight          0.0722      0.023      3.125      0.002       0.027       0.117
TouchScreen    -0.0243      0.035     -0.686      0.493      -0.094       0.045
Ips             0.0974      0.026      3.763      0.000       0.047       0.148
Ppi             0.0033      0.000      9.474      0.000       0.003       0.004
HDD          3.119e-05   2.83e-05      1.103      0.270   -2.43e-05    8.67e-05
SSD             0.0011   9.42e-05     11.457      0.000       0.001       0.001
==============================================================================
Omnibus:                       16.452   Durbin-Watson:                   1.992
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               20.448
Skew:                          -0.180   Prob(JB):                     3.63e-05
Kurtosis:                       3.505   Cond. No.                     4.21e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.21e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

From the summary we can see the some features have a p>0.05 so we can remove it from the formula.

[17]:
```python
model = smf.ols('Price ~ Ram + Weight + Ips + Ppi + SSD', data=df).fit()

# Print the summary of the model
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:                  Price   R-squared:                       0.603
Model:                            OLS   Adj. R-squared:                  0.601
Method:                 Least Squares   F-statistic:                     384.5
Date:                Mon, 24 Jun 2024   Prob (F-statistic):          5.80e-251
Time:                        12:11:57   Log-Likelihood:                -608.78
No. Observations:                1273   AIC:                             1230.
Df Residuals:                    1267   BIC:                             1260.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      9.5757      0.066    144.627      0.000       9.446       9.706
Ram            0.0461      0.003     14.193      0.000       0.040       0.052
Weight         0.0842      0.021      3.990      0.000       0.043       0.126
Ips            0.0952      0.026      3.687      0.000       0.045       0.146
Ppi            0.0032      0.000      9.759      0.000       0.003       0.004
SSD            0.0010   8.35e-05     12.348      0.000       0.001       0.001
==============================================================================
Omnibus:                       19.005   Durbin-Watson:                   1.994
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               23.545
Skew:                          -0.205   Prob(JB):                     7.72e-06
Kurtosis:                       3.526   Cond. No.                     1.82e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.82e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Although the R-squared did not increase after applying the new formula, the less features included results a more simple model and avoid overfitting.

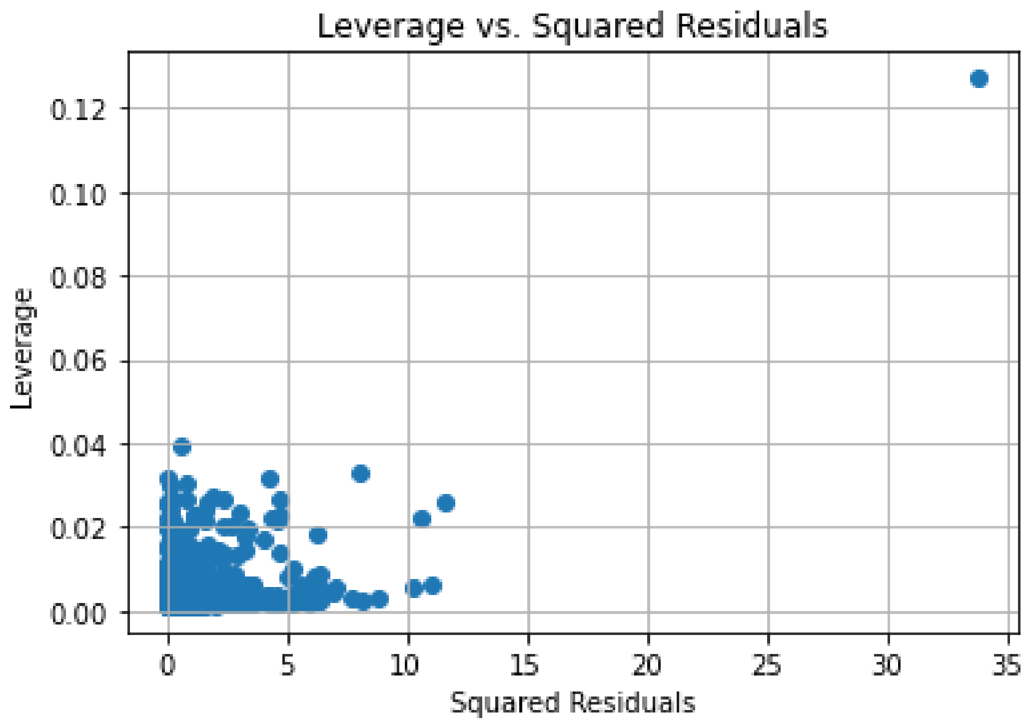
Leverage vs. the Square of the Residual

[18]:
```
formula = 'Price ~ Ram + Weight + Ips + Ppi + SSD'
model = smf.ols(formula, data=df).fit()

# Calculate influences and residuals
influence = model.get_influence()
resid_squared = influence.resid_studentized_external ** 2
leverage = influence.hat_matrix_diag

# Plot leverage vs. squared residuals
plt.scatter(resid_squared, leverage)
plt.ylabel('Leverage')
plt.xlabel('Squared Residuals')
```

```
plt.title('Leverage vs. Squared Residuals')
plt.grid(True)
plt.show()
```



2 Conclusion

In this project, I aimed to identify the key factors influencing laptop prices using machine learning techniques. The process involved multiple steps, including data loading, cleaning, exploratory data analysis, and building linear regression model.

2.1 Key Findings

2.1.1 Data Preparation and Cleaning

We started by loading the dataset from Kaggle, ensuring data types were appropriate, and removing non-numeric and less relevant features. This step was crucial to prepare the data for analysis and modeling.

2.1.2 Correlation Analysis

Through the correlation matrix and visualizations like heat maps and pair plots, we can identified that RAM, SSD, and PPI had the highest correlations with the laptop prices. This guided us in selecting these features for further analysis.

2.1.3 Simple Linear Regression

We performed simple linear regression for each of the highly correlated features (RAM, SSD, and PPI). The RAM feature showed the highest adjusted R-squared value of 0.486, indicating a moderate level of predictive power.

2.1.4 Multi-Linear Regression Model

To improve the predictive accuracy, we built a multi-linear regression model using multiple features. The model initially included RAM, Weight, TouchScreen, Ips, Ppi, HDD, and SSD. We refined the model by removing features with high p-values (>0.05), leading to a final model with RAM, Weight, Ips, Ppi, and SSD. This simplified model reduced the risk of overfitting while maintaining a balance between complexity and predictive power.

2.2 Implications

The analysis highlighted that certain hardware specifications, particularly RAM and SSD, significantly impact laptop prices. These findings can assist manufacturers and consumers in understanding price determinants and making informed decisions. However, the predictive power of our models, while reasonable, suggests that other factors not included in the dataset might also play significant roles in determining laptop prices.

2.2.1 Future work could involve:

Expanding the dataset to include more features, such as brand reputation, build quality, and market trends. Utilizing advanced machine learning techniques like ensemble methods to enhance predictive accuracy. Conducting time-series analysis to understand how the impact of different features on price evolves over time.

3 GitHub Link

Here is the link of the GitHub Repository: https://github.com/chloefung/DTSA5509-Final-Project/