CS6140 Project 1: Collecting and Analyzing Data

Description

My project is revolving around the diamond price dataset, where the price of the diamond is determined by multiple factors, such as carat, cut, color, clarity, depth, and table size. I used PCA and several regressions methods to find the correlations between these independent variables to the pricing of diamond.

Required items (file overview)

- diamonds.csv - the original dataset
- diamonds test set.csv - the split 25% test part
- diamonds train set.csv - the split 75% training part
- pcatestdata.csv - the data used to test the correctness of PCA
- CS6140 Project1 Report - the PDF report file
- multiple images:
  - 3rd degree price vs carat.png - the 3rd degree polynomial regression fit for carat vs price data
  - price vs carat.png - the linear regression between price and carat
  - price vs clarity.png - the linear regression between price and clarity
  - price vs color.png - the linear regression between price and color
  - price vs cut.ong - the linear regression between price and cut
  - price vs depth.png - the linear regression between price and depth
  - price vs table.png - the linear regression between price and table size
  - main.py - the commented code used to generate all the required plots and analysis

Extension appropriate result

Please refer to the extension section towards the end of this report.

Reflection

Some of the important knowledge I have learned are:

1. Data with a lot of features can usually be reduced to less features and become easier for data processing and analysis. However, data cleaning is also very important and should always be performed to make sure that the analysis is performed on legitimate data instead of yielding garbage results.
2. PCA analysis can reduce feature dimensionality but when multiple features are inter-correlated, it is hard to completely segregate one of the significant features among all. From the eigenvectors, we can also determine whether features are intercorrelated by checking if each eigenvector is significantly dominated by one feature.
3. Multiple linear regression is a very efficient and powerful tool to determine the relationship between all independent variables and dependent variable. It can be also used as a baseline to determine the performance of other fitting algorithms.

Consultation

Consulted with TA Yiming Ge about PCA and asked some questions during the lecture.

Task 1 Collect a Data Set

The submitted CSV data file is named diamonds.csv in the zipped folder

Task 2 Organize Data

The split data function is on line 66.

```
"/Users/haotianshen/Desktop/CS6140 Project 1/venv/bin/python" /Users/haotianshen/Desktop/CS6140 Project 1/main.py diamonds.csv 25
Test set: 24.53%, number of samples: 13230.
Training set: 75.47%, number of samples: 40710.
Test set file name: diamonds test set.csv.
Training set file name: diamonds train set.csv.
```

Figure 1: running the program with a test set percentage of 25%, the diamonds.csv is split into a training and a test set, where the test set is 24.53% and the training set is 75.47% percent. The split is completely random, and the file is named as diamonds train set.csv and diamonds test set.csv.

Task 3 & 4 Simple Linear Regression

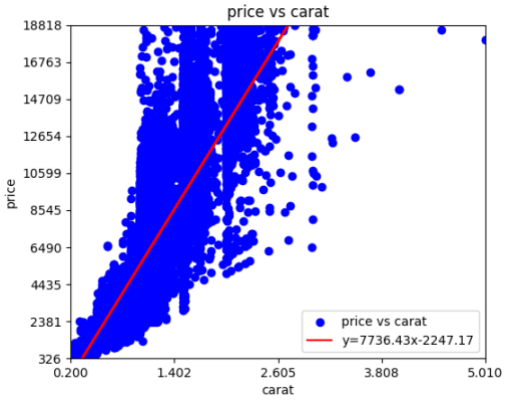

Figure 1: price vs carat linear fit, $r^2 = 0.846$, the slope is 7736.43. The slope is large, and the r^2 is close to 1, which means that the carat has a strong correlation with the price of the diamond, and it has a fairly big impact on the price.

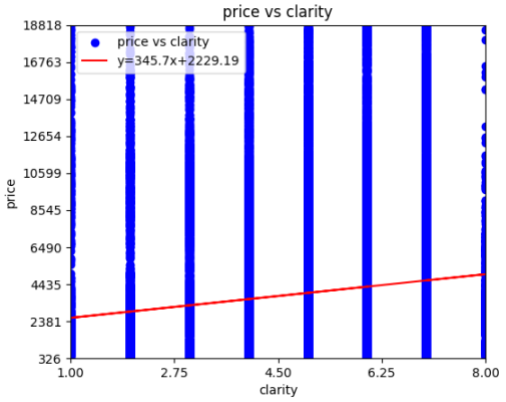

Figure 2: price vs clarity linear fit, $r^2 = 0.021$, the slope is 345.70. The slope is small and the r^2 is low, which means that the clarity of the diamond does not strongly affect the price of the diamond and it has weak correlation with the price.

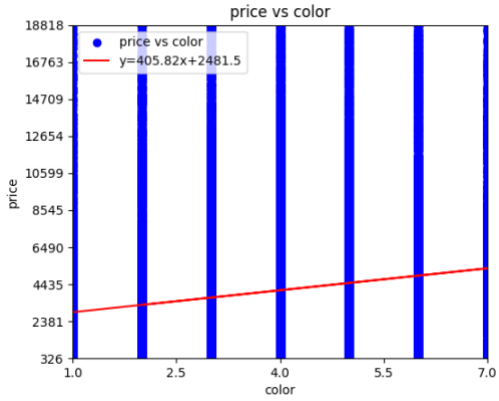

Figure 3: price vs color linear fit, $r^2 = 0.03$, the slope is 405.82. The slope is just a bit larger than that of the cut, and the r^2 is close to 0 as well, which means that the color of the diamond has almost no impact on the price of the diamond.

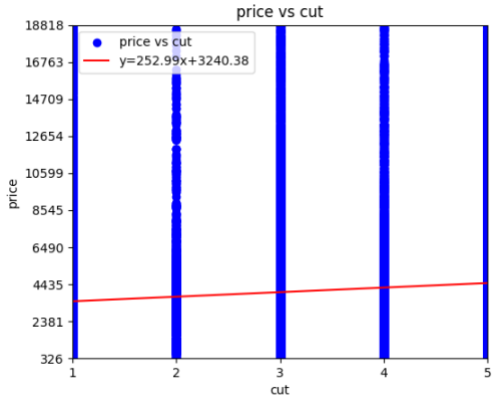

Figure 4: price vs cut linear fit, $r^2 = 0.011$, the slope is 252.99. The slope is small, the r^2 is close to 0, which means that the cut of the diamond has almost no impact on the price of the diamond, and it has no strong correlation at all.

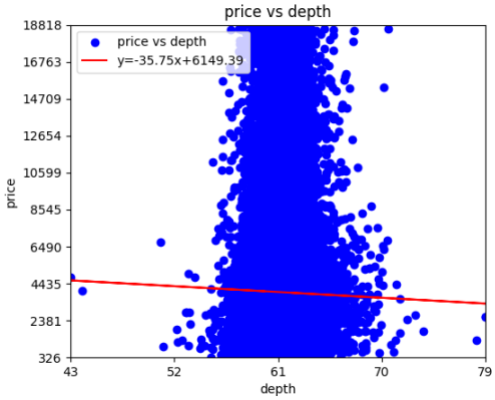

Figure 5: price vs depth linear fit. $r^2 = 0$, the slope is -35.75. The r coefficient indicates that the price and depth are not correlated.

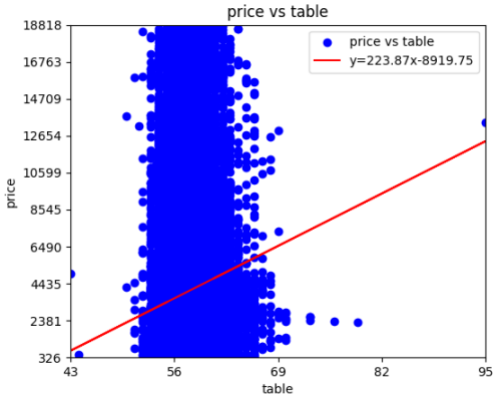

Figure 6: price vs table linear fit. $r^2 = 0.016$, the slope is 223.87. The r coefficient is also very low, which means the correlation is very low even though table and price are positively correlated based on the slope.

Task 5 Multiple Linear Regression

Coefficient: [8793.25228914, 9.23258284, -322.15638338, -535.59786213, -79.88919037, -63.2496326]
Intercept: 9261.219264558018
r-coefficient: 0.9026675203925671

The r-coefficient is 0.903, which means the multiple linear regression fits the data well. Among all the independent variables, 'carat' and 'clarity' are the two most strongly correlated variables to the dependent variable 'price'. 'Depth', 'table', and 'cut' are not as important. 'Color' is somewhat correlated to the price but not as greatly related as 'carat' and 'cut'. 'Cut' and 'carat' have positive correlations, whereas all other independent variables have negative correlations with the price of the diamond.

Task 6 Polynomial Model

Coefficient: [-278.13216191 1113.21203664 5527.93657189 -1239.32005577]

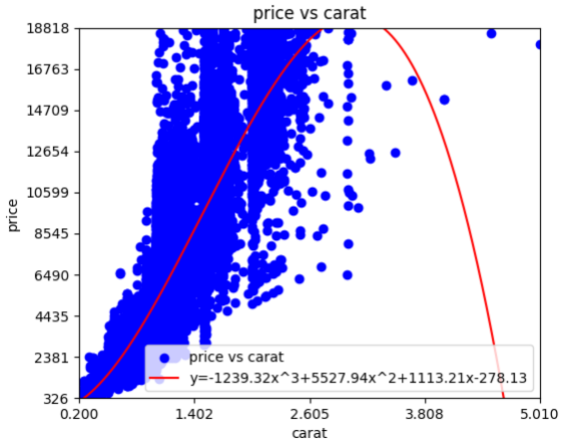

Figure: 3rd order polynomial model for price vs carat data.

Task 7 PCA

whitened data
mean is
[[2.2 3.48 4.36]]
standard deviation is
[[1.16619038 1.72904598 0.5425864]]
data after whitening:
[[-1.02899151 -1.0873048   0.25802342]
 [0.68599434 0.53208533 0.62662831]
 [1.54348727 1.57312184 -0.84779125]
 [-1.02899151 -1.02946944 -1.40069858]
 [-0.17149859 0.01156707 1.36383809]]
U is
[[-0.47074991 -0.13790587 0.26704592]
 [0.27931217 -0.26622668 0.69194235]
 [0.68820291 0.41253274 -0.20564651]
 [-0.47572392 0.60402963 -0.12782367]
 [-0.02104125 -0.61242981 -0.62551809]]
S is
[3.15833141 2.23506254 0.17157545]
eigenvector is
[[0.70649998 0.70689017 0.03411832]
 [0.03557058 0.01268   -0.99928672]
 [0.70681858 -0.70720966 0.01618608]]
eigenvalue is:
[2.49376433 1.24887614 0.00735953]
The eigen sum is 3.7499999999999996
The number of features to keep are: 2
projected data:
[[-1.48678424 -0.30822824]
 [0.88216039 -0.59503329]
 [2.17357288 0.92203647]
 [-1.5024938   1.35004399]
[-0.06645523 -1.36881893]]

non-whitened data
mean is
[[2.2 3.48 4.36]]
standard deviation is
[[1. 1. 1.]]
data after whitening:
[[-1.2 -1.88 0.14]
 [0.8   0.92 0.34]
 [1.8   2.72 -0.46]
 [-1.2 -1.78 -0.76]
 [-0.2   0.02 0.74]]
U is
[[-0.4784822   0.12790041 -0.25813428]
 [0.26034743 0.266788   -0.69908478]
 [0.69948526 -0.39902302 0.19390201]
 [-0.46219863 -0.61152516 0.14111464]
 [-0.01915187 0.61585977 0.62220241]]
S is
[4.65744808 1.21376789 0.23440331]
eigenvector is
[[0.55824536 0.82963815 0.00791549]
 [-0.03924415 0.01687435 0.99908716]
 [-0.82874725 0.55804641 -0.04197848]]
eigenvalue is:
[5.42295565 0.36830813 0.01373623]
The eigen sum is 5.805000000000003
The number of features to keep are: 2
projected data:
[[-2.22850599 0.15524141]
 [1.21255465 0.32381871]
 [3.2578163 -0.48432134]
 [-2.15266611 -0.7422496]
 [-0.08919885 0.74751082]]

The eigenvalues for the whitened data are significantly smaller than the eigenvalues for the non-whitened data. This is because data whitening involves normalizing the data so that it has zero mean and unit variance, and then multiplying it by the inverse square root of the covariance matrix. This step scales down the data, which in turn leads to smaller eigenvalues. The difference in magnitude between the first two eigenvalues in the whitened data is much less than the difference in magnitude between the first two eigenvalues in the non-whitened data. This suggests that the data is more decorrelated and the variance is more evenly distributed among the principal components after whitening. The third eigenvalue is much smaller than the other two in both cases. This suggests that the third principal component explains very little of the variance in the data.

Task 8 Apply PCA to Diamond Dataset

whitened diamond data
mean is
[[0.79996288 2.77206913 3.59814646 4.95378451
61.75185968 57.4500676]]
standard deviation is
[[0.47268742 1.64015278 1.69808719 1.64843891
1.44068083 2.23577474]]
data after whitening:
[[-1.24810363 1.35836789 -0.94114512 0.63467047
-1.35481755 1.58778625]
 [-1.20579237 0.74866859 -0.94114512 -0.57859864
-3.36775472 3.37687525]
 [-1.07885859 1.35836789 1.41444653 0.02803591
0.44988474 0.2459695]
 ...
 [-0.16916651 0.74866859 -1.53004303 0.63467047
0.93576612 -1.09584725]
 [-0.21147777 0.13896929 -1.53004303 0.63467047
0.72753124 1.140514]
 [-0.10569962 -1.0804293 -1.53004303 1.24130502
0.31106148 -1.09584725]]

U is
[[0.0052816 0.01022548 0.00353692 0.00039388
0.00585871 -0.00077891]
 [0.00766067 0.01845369 -0.00394192 -0.00537935
0.001673   -0.00891792]
 [0.00205465 0.00015777 -0.0033428   0.00868165
0.00930518 0.00235777]
 ...
 [-0.00181142 -0.00062333 0.00948478 0.0027905
-0.00161969 0.00777598]
 [0.0016996   0.00259771 0.00831709 0.00204839
-0.00279225 -0.00697487]
 [-0.00398895 -0.00188191 0.00941362 -0.0060934
0.00048002 0.00057176]]
S is
[274.60337926 232.91468854 204.46323005
183.53700251 146.17318091
 132.51448754]
eigenvector is
[[0.42585689 0.5356283   0.16749187 0.3770717 -0.20167402 0.56642672]
[-0.50153514 0.21527856 -0.44599316 -0.32907241 -0.53158073 0.33517312]
[-0.08541677 0.07952873 -0.73432951 0.57377762 0.33893799 -0.05513267]
[-0.30628682 0.53772008 0.14059627 -0.3303142 0.68378277 0.14356824]
[-0.67019922 0.04106259 0.46003052 0.55667793 -0.13904719 -0.09107338]
[0.12974503 0.60795035 -0.04900368 -0.0361903 -0.27382031 -0.73135077]]
eigenvalue is:
[1.85375426 1.33362634 1.02771062 0.82810933 0.52526178 0.43168517]
The eigen sum is 6.000147499877257
The number of features to keep are: 6
projected data:
[[1.45034492 2.38166523 0.72317056 0.07229201
0.85638624 -0.10321649]
 [2.10364676 4.298136   -0.80597693 -0.9873103
0.24454817 -1.18175368]
 [0.56421439 0.03674597 -0.68347949 1.59340394
1.36016735 0.31243842]
 ...
 [-0.49742301 -0.14518235 1.93928945 0.51216049
-0.23675505 1.03043059]
 [0.46671658 0.60504486 1.70053965 0.37595535
-0.40815241 -0.92427156]
 [-1.09538037 -0.43832454 1.92473967 -1.11836352
0.07016549 0.07576694]]
whitened projected data correlation
[1296.73699694 -1533.04743993 -489.91442418 -1149.20695459 -3499.30242737   742.18299029]
3941.684063030265 0.9026675203925671

Whitened data eigenvalues: [1.85375426 1.33362634 1.02771062 0.82810933 0.52526178 0.43168517]
Independent variable columns: 'carat', 'cut', 'color', 'clarity', 'depth', 'table'
Dependent variable column: 'price'

After applying PCA to my data and keeping a 0.95 cumulative sum of eigenvalues, the number of significant features to keep is still 6, which is not different from the dataset used in the previous multiple linear regression. All variables are correlated to some degree, which is understandable since all factors contribute to the pricing of diamonds. There are a total of 6 eigenvectors sorted based on eigenvalues.

All independent variables are somewhat correlated because depth, table size, and cut contribute to how heavy the diamond is, which is the same as how many carats it has. 'Color' and 'clarity' are probably the least correlated independent variables in the dataset.

It makes sense to use data whitening to make the data zero-mean and unit variance because they are measured in different units. For example, carat is usually measured in the range from 0 to 5, but other independent variables, such as color and clarity, are using dummy coding, and depth and table are measured in millimeters. To avoid large unit feature domination, whitening my diamond data can help balance the scales and make it easier to compare the features. Whitening the data can also reduce inter-correlation between independent variables and fit the data better. Without whitening, the larger feature will dominate and make the justification of the principal component difficult.

Task 9 Implement Multiple Linear Regression on Non-Whitened Projected Data

Refer to the result from Task 8:
Eigenvalues after whitening: [1.85375426 1.33362634 1.02771062 0.82810933 0.52526178 0.43168517]

Eigenvector is
[[0.42585689 0.5356283   0.16749187 0.3770717 -0.20167402 0.56642672]
[-0.50153514 0.21527856 -0.44599316 -0.32907241 -0.53158073 0.33517312]

[-0.08541677 0.07952873 -0.73432951 0.57377762 0.33893799 -0.05513267]
[-0.30628682 0.53772008 0.14059627 -0.3303142   0.68378277 0.14356824]
[-0.67019922 0.04106259 0.46003052 0.55667793 -0.13904719 -0.09107338]
[0.12974503 0.60795035 -0.04900368 -0.0361903 -0.27382031 -0.73135077]]

Coefficient: [1296.73699694 -1533.04743993 -489.91442418 -1149.20695459 -3499.30242737   742.18299029]
Intercept: 3941.684063030265
r coefficient: 0.9026675203925671

Coefficient: [8793.25228914, 9.23258284, -322.15638338, -535.59786213, -79.88919037, -63.2496326]
Intercept: 9261.219264558018
r-coefficient: 0.9026675203925671

And the number of features to keep achieving 0.95 cumulative sum is 6, which means that all features are kept in order to achieve 95% description of the dataset. This means that all features are correlated to the dependent variable (price) and should be kept. Thus, the r-coefficient is the same as the previous multiple linear regression without data projection onto the eigen space. However, the coefficients become more evenly distributed due to normalization.

Each eigenvector represents a direction or linear transformation of the data where the variance of the data is maximized. Each eigenvector is associated with a scalar eigenvalue, which represents the amount of variance explained by that eigenvector. Additionally, each eigenvector represents a principal component of the data, which can be used for dimensionality reduction. The first eigenvector is most strongly related to the dependent data because its eigenvalue is the largest.

Based on the eigenvalues and eigenvectors, we can tell that:
First eigenvector: 'cut' and 'table' are the top two contributing variables to maximum variance in this direction.
Second eigenvector: 'carat' and 'depth' are the top two contributing variables to maximum variance in this direction.
Third eigenvector: 'color' and 'clarity' are the top two contributing variables to maximum variance in this direction.
Fourth eigenvector: 'depth' and 'cut' are the top two contributing variables to maximum variance in this direction.
Fifth eigenvector: 'carat' and 'clarity' are the top two contributing variables to maximum variance in this direction.
Sixth eigenvector: 'cut' and 'table' are the top two contributing variables to maximum variance in this direction.

Extension

Extension 1: Make your data organization program repeatable if the user provides a seed for the random number generator. Be sure to demonstrate that the program is, in fact, repeatable.

The logic for this to work was: No matter what seed the random generator will use, I will always produce a a number between 1 and 100, inclusive, and based on whether it is greater than the test percentage, I will assign that line in the original data file to training set file or test set file. This way, it will always approximately maintain the desired test set percentage.

Result:
using seed 12300
Test set: 24.5%, number of samples: 13214.
Training set: 75.5%, number of samples: 40726.
Test set file name: diamonds test set.csv.
Training set file name: diamonds train set.csv.

using seed 18
Test set: 24.52%, number of samples: 13226.
Training set: 75.48%, number of samples: 40714.
Test set file name: diamonds test set.csv.
Training set file name: diamonds train set.csv.

using seed 1234
Test set: 24.53%, number of samples: 13229.

Training set: 75.47%, number of samples: 40711.
Test set file name: diamonds test set.csv.
Training set file name: diamonds train set.csv.

Extension 2: Use both Ridge and Lasso regression on your data and discuss the differences in the results.
Using multiple linear regression:
Coefficient: [8793.25228914, 9.23258284, -322.15638338, -535.59786213, -79.88919037, -63.2496326]
Intercept: 9261.219264558018
r-coefficient: 0.9026675203925671

using Ridge regression
[8.80448337e+03 8.68272338e+00 -3.16520830e+02 -5.26028016e+02
 -7.69833290e+01 -5.90654035e+01]
using alpha=0.1 Ridge regression
training set: 0.9041712941054072
test set: 0.9018708862096633

[8.80333864e+03 8.69027645e+00 -3.16425359e+02 -5.25912793e+02
 -7.69718863e+01 -5.90378620e+01]
using alpha=1 Ridge regression
training set: 0.9041712767265979
test set: 0.9018721029378707

[8.79190785e+03 8.76570422e+00 -3.15472071e+02 -5.24762267e+02
 -7.68576212e+01 -5.87628486e+01]
using alpha=10 Ridge regression
training set: 0.9041695434354329
test set: 0.9018826789117499

[8679.22474438   9.50982857 -306.07929309 -513.42476726 -75.73097385
  -56.05232417]
using alpha=100 Ridge regression
training set: 0.9040007065357023
test set: 0.901833819511204

using Lasso regression
[8.80394390e+03 8.61040209e+00 -3.16444141e+02 -5.25940815e+02
 -7.69169659e+01 -5.89964174e+01]
using alpha=0.1 Lasso regression
training set: 0.9041712878498588
test set: 0.9018708091195856

[8.79770454e+03 7.92572371e+00 -3.15635054e+02 -5.25014626e+02
 -7.62979382e+01 -5.83245925e+01]
using alpha=1 Lasso regression
training set: 0.9041706066350003
test set: 0.9018709364791403

[8.73482293e+03 1.07788900e+00 -3.07489311e+02 -5.15677870e+02
 -7.01403654e+01 -5.16036153e+01]
using alpha=10 Lasso regression
training set: 0.9041013379027913
test set: 0.9018103801435283

[8096.98794664   -0.   -226.66809974 -427.58817741 -12.55125178
  -10.08974405]

using alpha=100 Lasso regression
training set: 0.8978039528466384
test set: 0.8956599673799337

Analysis of Extension 2:
Ridge Regression is a form of regularization that adds a penalty term to the cost function. The penalty term is the sum of the squares of the weight coefficients (L2 regularization term). The Ridge Regression cost function is:

cost = MSE + (alpha * (sum of square of weight coefficients))

Lasso Regression is also a form of regularization that adds a penalty term to the cost function. However, the penalty term is the sum of the absolute values of the weight coefficients (L1 regularization term). The Lasso Regression cost function is:

cost = MSE + (alpha * (sum of absolute value of weight coefficients))

Ridge Regression uses the L2 regularization term, while Lasso Regression uses the L1 regularization term. As a result, Ridge Regression will shrink all the weight coefficients towards zero but will not set any of them exactly to zero. In contrast, Lasso Regression can set some of the weight coefficients to exactly zero, effectively selecting a subset of the input features for the model.

When setting the Ridge alpha=0.1, the coefficient results are very similar to the one shown in the original multiple linear regression. This is since setting alpha=0 for Ridge regression will basically turn it into a multiple linear regression. As the value of alpha increases, the cost function starts to penalize the magnitude of the weight coefficients and shrink them towards zero. However, using alpha=100 yields a slightly poorer r-coefficients on both the training and test sets compared to those of alpha=10, so we can be certain that all features play an important role in determining the price of diamond.

When setting the Lasso alpha=0.1 or 1, the coefficient results are very similar to the one shown in the original multiple linear regression. However, when using a larger alpha, for example, 10 or 100, the Lasso algorithm tends to penalize the coefficients harder than the Ridge regression. We can see from alpha=100 that “cut” coefficient was completely 0. Thus, Lasso regression is helpful in feature selection but does not fit my data selection that well compared to Ridge regression. Moreover, we can tell from the result using alpha=100 that most variable still have a fairly large coefficients, which indicate that some of the variables may be inter-correlated.