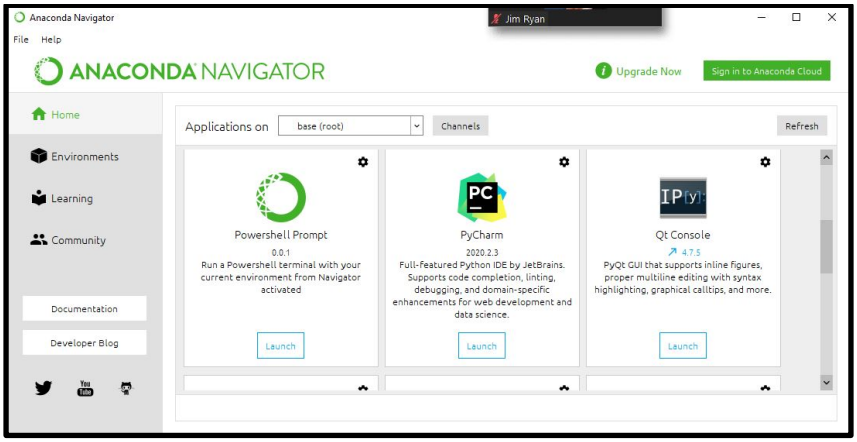
Data Mining Fundamentals

What is data mining?

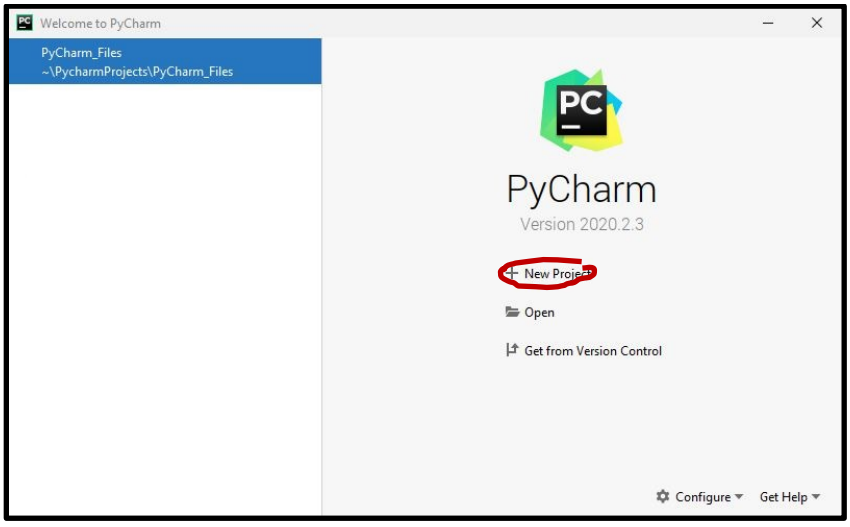
Data mining is a combination of several different processes, tools, or methodologies used to extract relevant data from a larger set of raw data. Data mining implies analyzing data patterns in large batches using one or more software applications. With respect to this tutorial, we will narrow our focus on Python data mining processes of cluster analysis, regression and classification, association and correlation, as well as dimensionality reduction. Python has built-in features to help in data mining methodologies for various data sets. For more on data mining not in this tutorial, refer to the URL: https://www.tutorialspoint.com/data_mining/index.htm

Create a Data Mining Project in PyCharm

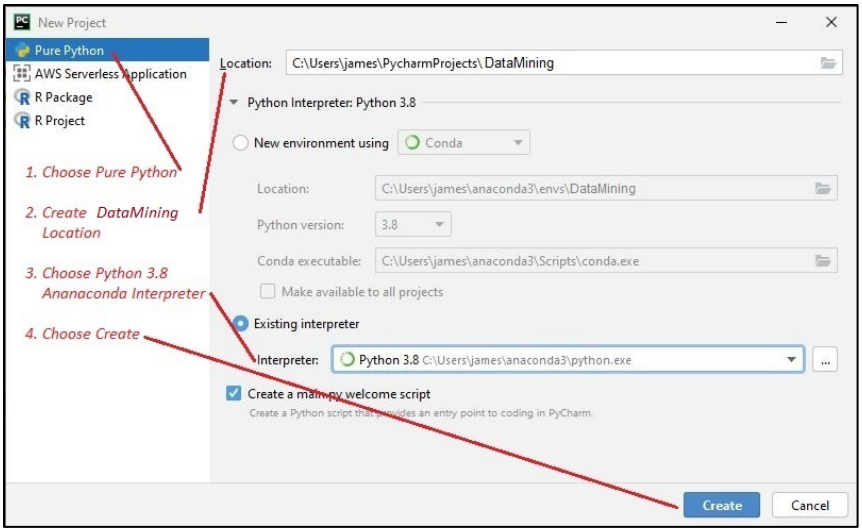
0) To ensure Python package libraries are available for import, use Anaconda Navigator to launch PyCharm



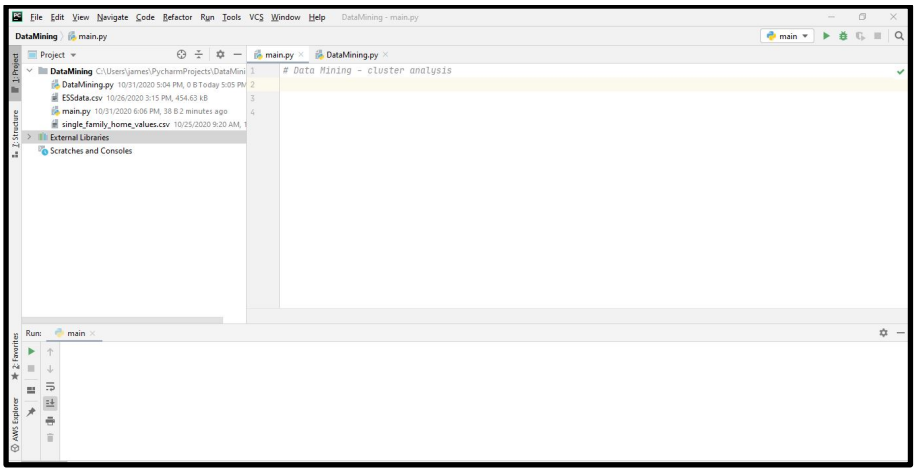
1) Open PyCharm and choose a New Project.



1) Choose a Pure Python project,
2) Change your project location to create a DataMining folder
3) Be sure to choose the Python interpreter as Python 3.8 associated with Anaconda.
4) Choose Create to open PyCharm with these settings.



5) *Download* the single_family_home_values.csv file from the Week 10 Canvas Module to the DataMining directory created in the last step. We used this CSV file during Week 09 Data Wrangling.



All visualizations may be viewed at 400% magnification for clarity.

Reading a CSV file into a Pandas DataFrame:

We learned about Pandas in Tutorial Seven. Again this tutorial, we will look at DataFrame or df. For more specific details about Pandas not covered in this tutorial, please refer to the following URL: https://www.tutorialspoint.com/python_data_science/python_pandas.htm

1) Anaconda will load the libraries in PyCharm for you. To access the four libraries we are using for Tutorial Eight (e.g., pandas, seaborn, matplotlib.pyplot, and KMeans), enter the following code into Main.py in the DataMining project directory of PyCharm:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

2) To format the run window to handle the size of the csv data file we will be data mining in Tutorial Eight, enter the following Python code:

```
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
```

3) To read the single_family_home_values.csv file into a pandas dataframe, type the following Python code into main.py via PyCharm in the line below the previous code above.

```
df = pd.read_csv(r'single_family_home_values.csv')    # reads Zillow file
```

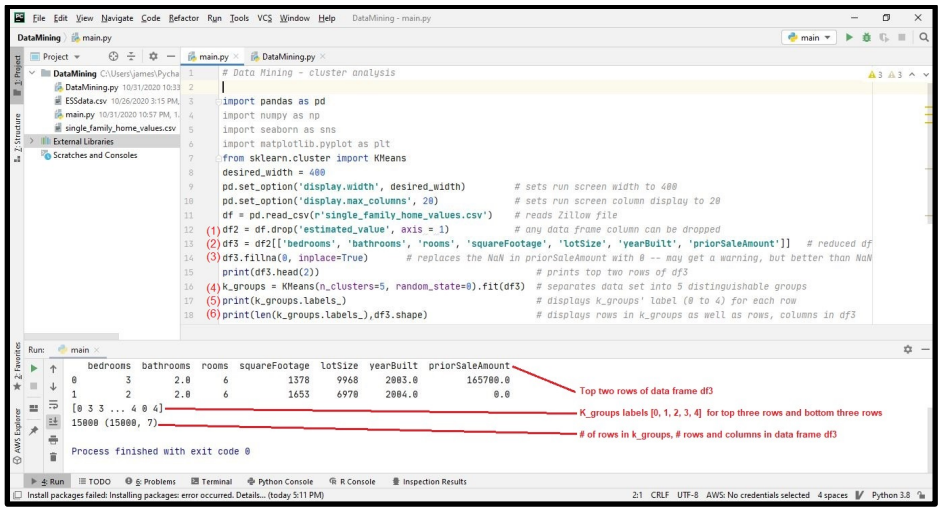
4) Once the csv file is loaded into a dataframe (df), we can begin data mining. Any of the following Python functions may be used with a df, once it has been assigned.

```
print(df.head(2))     # returns top two rows
print(df.tail(5))     # returns bottom five rows
print(type(df))       # returns type of df
print(df.shape)       # returns row & column count
print(df.describe()) # returns data profiling metrics
print(df.info())      # returns column attributes
```

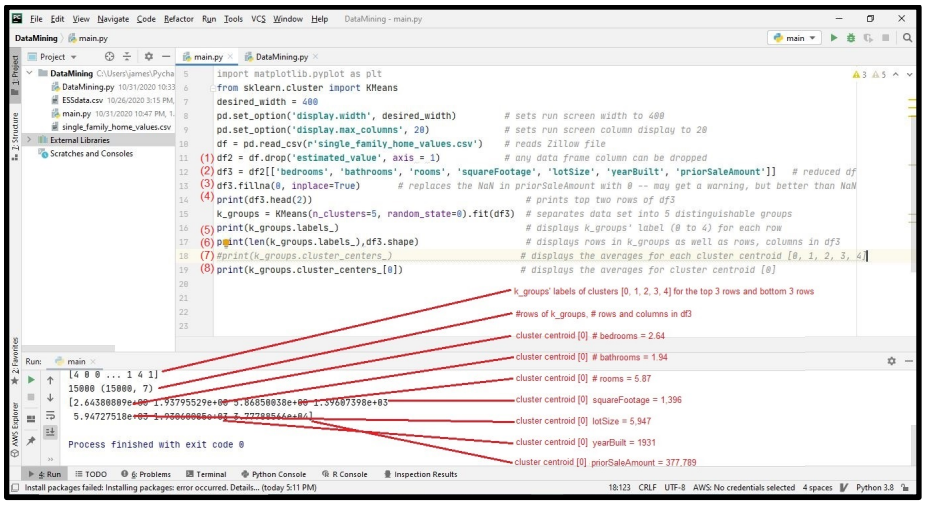
Cluster Analysis:

Cluster analysis is a type of unsupervised learning method and a common data mining technique for statistical data analysis. Cluster analysis is a task of dividing the data set of observations into subsets, called clusters, in such a way that observations in the same cluster are similar in one sense and dissimilar to observations in other clusters. The main goal of clustering is to group the data on the basis of similarity and dissimilarity. For cluster analysis in Tutorial Eight, we will use a well-known algorithm for data clustering the Scikit-learn package library called KMeans. KMeans takes the number of K-groups assigned, so it assums that a known K of clusters exist. This is also called flat clustering. KMeans is an iterative algorithm, where data points are randomly assigned to each K-group and the K-groups' centroids in each iteration find optimized separation until the K-groups' centroids reach their optimal location separation. For more information and details on Cluster Analysis, refer to the following URLs:
https://www.tutorialspoint.com/artificial_intelligence_with_python/artificial_intelligence_with_python_unsupervised_learning_clustering.htm
https://www.tutorialspoint.com/data_mining/dm_cluster_analysis.htm

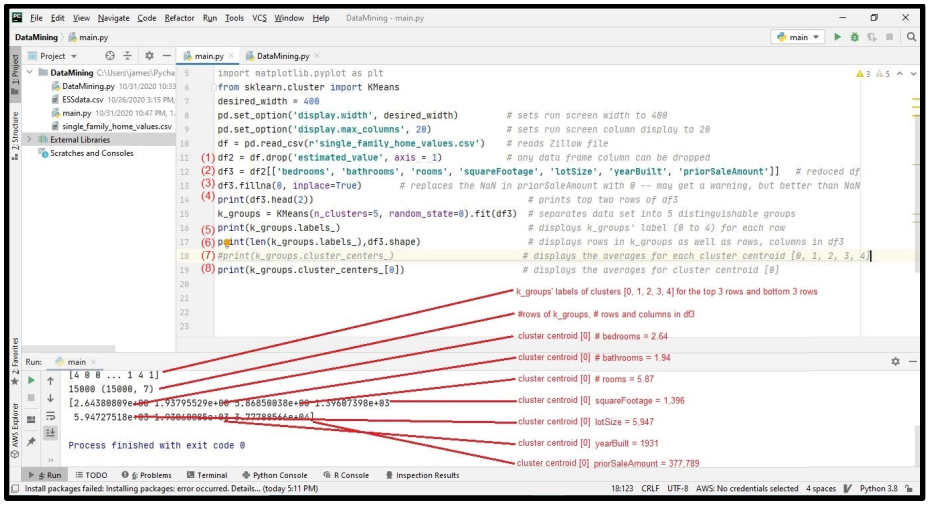
1) We imported pandas, numpy, seaborn, matplotlib.pyplot, and KMeans from sklearn.cluster. Looking at the PyCharm screenshot below, you will see how to (1) create a new data frame df2 that dropped a column (estimated_value) from a data frame, (2) slice the new df2 into a new data frame df3 that contains seven columns, (3) replace the df3 fields in columns with missing data with 0, (4) uses KMeans to create a new data frame k_groups that separates all rows of df3 data observations into 5 similar cluster centroids, (5) displays the k_groups' labels [0 to 4] for each of the top 3 rows and bottom 3 rows, and (6) compare the # rows in k_groups to the # of rows and columns in the reduced data frame df3.



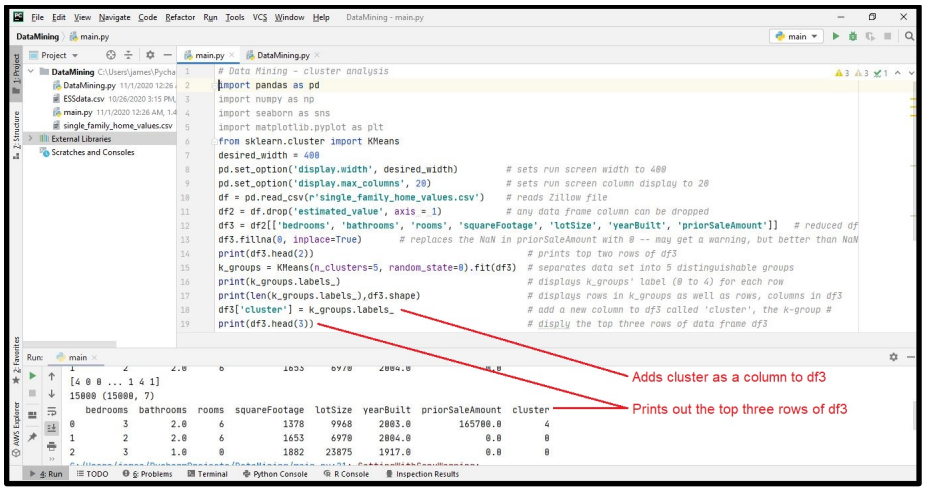
2) In the following PyCharm screenshot, we (7) print out the cluster centroid coordinates for each of the seven k_groups columns. Each cluster centroid show coordinates respectively for bedroom, bathroom, rooms, squareFootage, lotSize, yearBuilt', and priorSaleAmount.



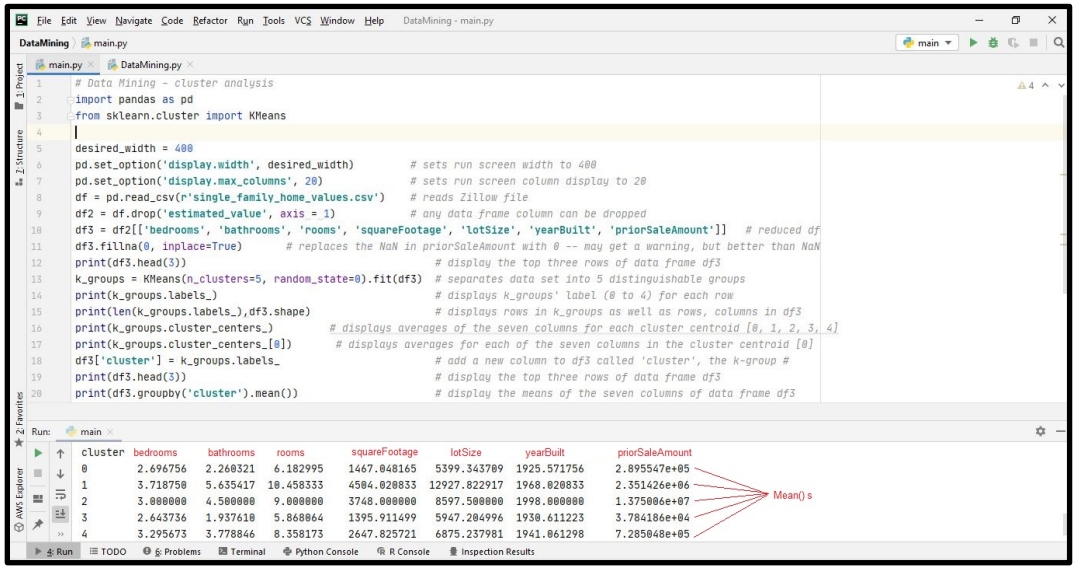
3) The PyCharm screenshot below identifies how to show the k_groups cluster centroid coordinates for k_groups cluster centroid [0].



4) The PyCharm screenshot below identifies how to add the k_groups cluster centroid number as a column of the reduced data frame df3.

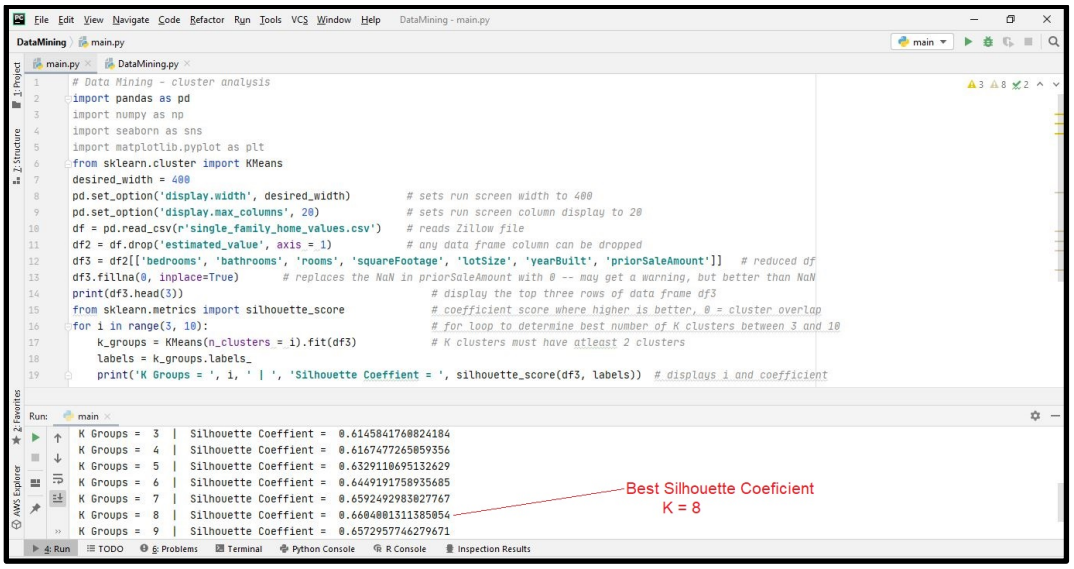


5) The PyCharm screenshot below depicts how to get k_groups means (i.e., fd3.groupby('cluster').mean()) from data frame df3. You can substitute means with minimum (i.e., fd3.groupby('cluster').min()), maximum (i.e., fd3.groupby('cluster').max()), or median (i.e., fd3.groupby('cluster').median()).



6) The PyCharm screenshot below depicts how to use a 'for loop' to calculate a silhouette_score library package from sklearn.metrics. The silhouette_score returns a silhouette coefficient (i.e., -1 < silhouette coefficient < 1) over all samples in the clusters. A silhouette coefficient near 1 is prefeered and near 0 reflects cluster overlap. More information on silhouette_score from sklearn.metrics is available via the URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html.

Note: In the screenshot below, df3 has dropped the 'cluster' column.



The DataMining.py file has the Python code for all screenshots in this Tutorial.

5) **Exercise deliverable:** Construct an example of a cluster analysis in Python code. Use either the single_family_home_values.csv or your data project csv. Determine the correct number of K clusters to use (i.e., see previous PyCharm screenshot in step 6) and then let Python calculate the means of all the columns separated into K clusters (i.e., steps 1 to 5). Capture two PyCharm screenshots for use later: (1) one of preferred K clusters due to silhouette coefficient and (2) a listing of dataframe.groupby('clusters').mean() for the preferred K clusters.

Classification & Regression:

Data mining to discover patterns in data that lead to actionable insights can be classified into two groups based on how the algorithms "learn" about data to make predictions: supervised and unsupervised learning. Practical machine learning uses supervised learning. Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output Y = f(X) . The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. For example, a classification algorithm will learn to identify animals after being trained on a dataset of images that are properly labeled with the species of the animal and some identifying characteristics. Supervised learning problems can be further grouped into Regression and Classification problems. Both problems have as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for regression (i.e., linear regression) and categorical for classification (i.e., logistic regression). For more discussion on the differentiation between classification and regression, refer to the URL: https://machinelearningmastery.com/classification-versus-regression-in-machine-

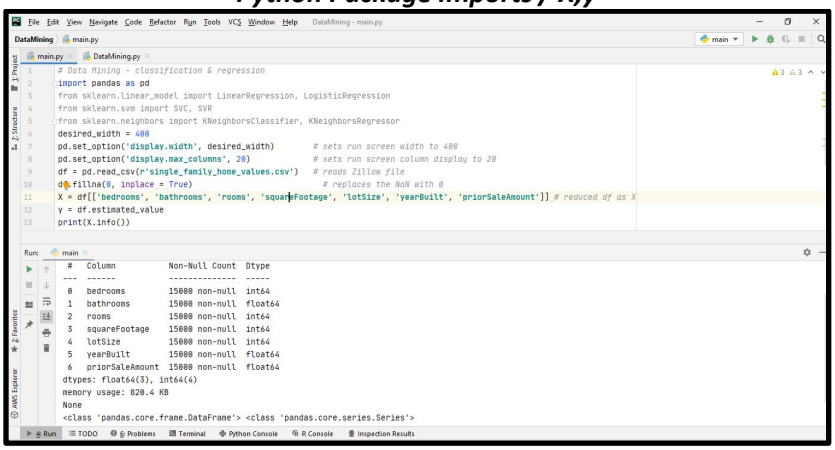
learning/#:~:text=Fundamentally%2C%20classification%20is%20about%20predicting,is%20about%20predicting%20a%20quantity.&text=That%20classification%20is%20the%20problem,quantity%20output%20for%20an%20example.

1) The following Python commands establish the PyCharm libraries and variables we will use in understanding classification and regression:

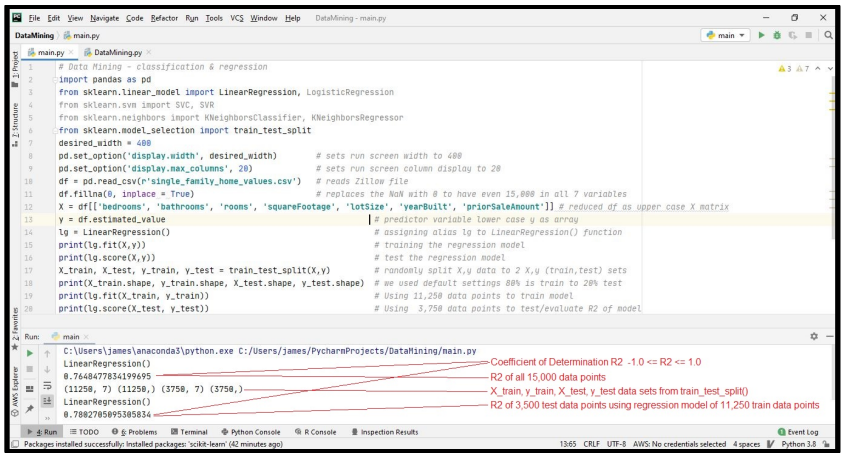
```
# Data Mining - classification & regression
import pandas as pd
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.svm import SVC, SVR
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.model_selection import train_test_split
desired_width = 400
pd.set_option('display.width', desired_width)    # sets run screen width to 400
pd.set_option('display.max_columns', 20)         # sets run screen column display to 20
df = pd.read_csv(r'single_family_home_values.csv')    # reads Zillow file
df.fillna(0, inplace = True)                          # replaces the NaN with 0
X = df[['bedrooms', 'bathrooms', 'rooms', 'squareFootage', 'lotSize', 'yearBuilt',
'priorSaleAmount']] # reduced df as X
y = df.estimated_value
```

Several of the library packages from sklearn have classifiers (i.e., categorical variables) and regressors (i.e., numeric variables) listed. Logistic regression is a classifier that has regression in the name. sklearn expects X to be a feature matrix (Pandas Dataframe) and y to be a response vector (Pandas Series). The train_test_split function randomly divides a data set into an 80% train and 20% test data sets. The following PyCharm screenshots demonstrate the Python code above using LinearRegression() as a regressor model and LogisticRegression() as a classifier model. Note a categorical variable was created for estimated_value in the classifier model. The DataMining.py file has the Python code for all screenshots in this Tutorial.

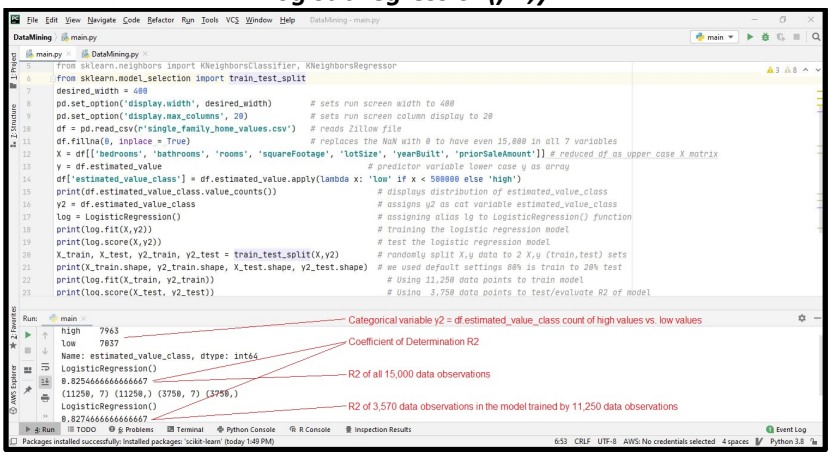
Python Package Imports / X,y



LinearRegression() X,y

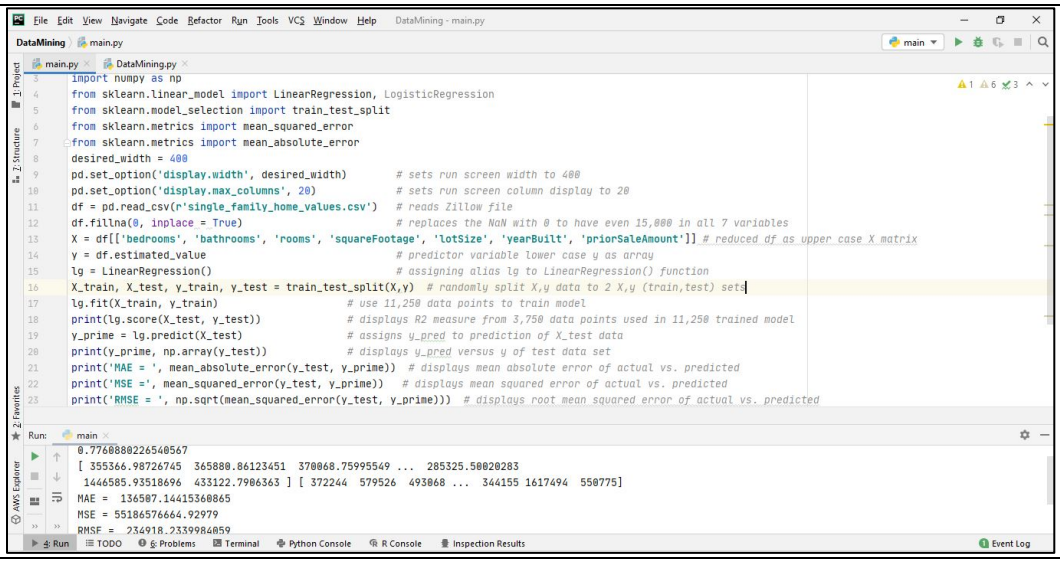


LogisticRegression() X,y2

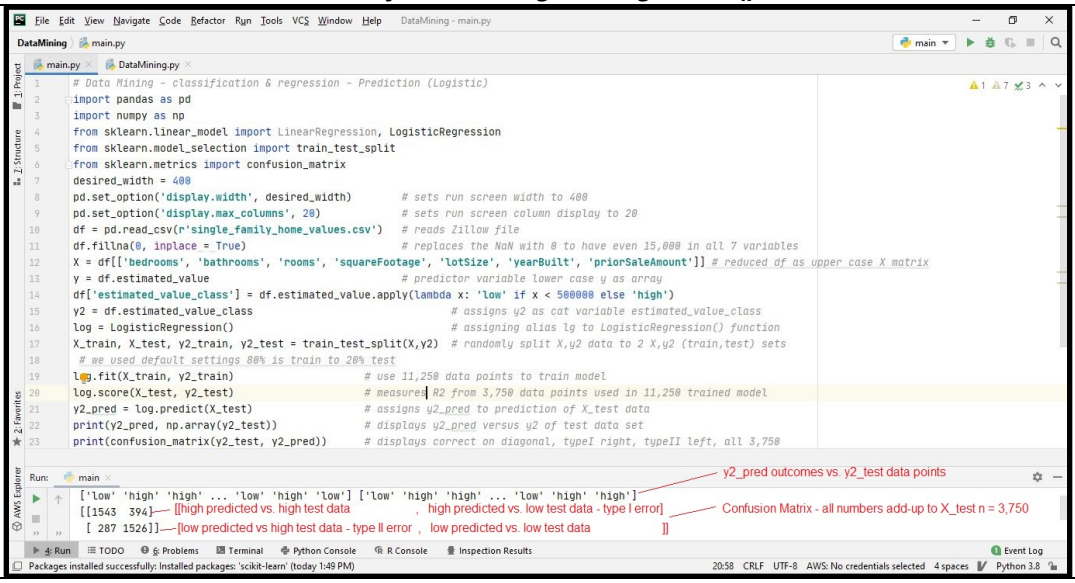


2) ***Prediction Check:*** When developing regression or classification models, besides using R2, you can use a Python confusion matrix on classifications to check what predictions would be and the associated Type I and Type II errors. For regressions, you can evaluate metrics (e.g., mean absolute error, mean squared error, and root mean squared error). The following two PyCharm screenshots show comparison metrics and a confusion matrix for regression and classification, respectively.

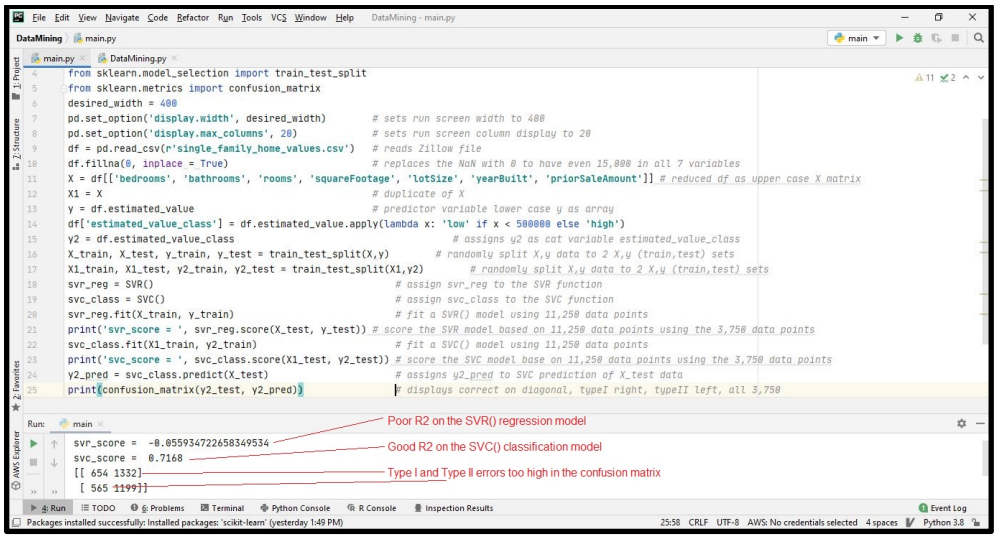
Regression – Linear Regression()



Classification – Logistic Regression()

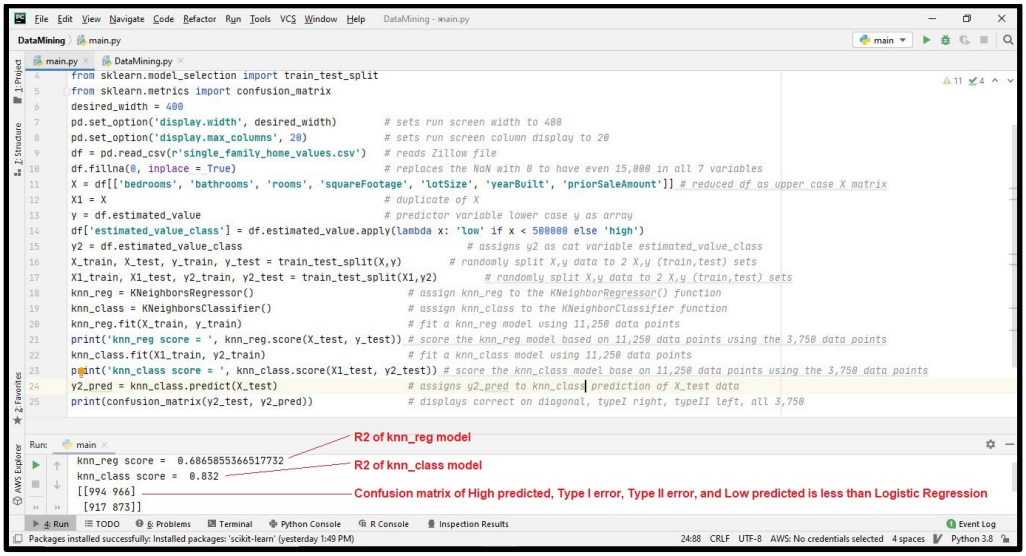


3) ***Support Vector Machine - Regression (SVR) / Classification (SVC):*** SVR.score() or SVC.score() return the coefficient of determination R2 for the regression or classification model. The same prediction check on classification can be run using the confusion matrix. For more detail on SVR, please refer to the following URL: https://scikit-learn.org/stable/modules/svm.html#svm-regression The following PyCharm screenshot depict Python code using SVR, SVC, and a predict check on classification.



4) ***K Nearest Neighbors (KNN) – Regressor & Classifier:*** Provides functionality for unsupervised and supervised neighbors-based learning methods. Supervised neighbors-based learning comes in two flavors: classification for data with discrete labels, and regression for data with continuous labels. The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point and predict the label from these. Being a non-parametric method, it is often successful in classification situations where the decision boundary is very irregular. The classes in sklearn.neighbors can handle either NumPy arrays or scipy.sparse matrices as input. A large number of possible distance metrics are supported for dense matrices, but fewer for sparse matrices. For more information on KNN, please refer to the following URL: https://scikit-learn.org/stable/modules/neighbors.html#unsupervised-neighbors

The following PyCharm screenshot demonstrates a KNN regression, classification, and classification prediction check example using Python code.



5) **Exercise deliverable:** Construct a regression, classification, and prediction check example using Python code. Use either the single_family_home_values.csv or your data project csv. If you are using the csv above, then mix-up the columns used for X. (1) Choose from Linear/Logistic, SVR/SVC, or K Nearest Neighbor methods of regression and classification. (2) Be sure to use the train_test_split() function in both regression and classification. (3) Choose either of the regression or classification example for the prediction check. (4) Capture PyCharm screenshots for each of the three examples (e.g., regressor, classifier, & prediction check) to use later in this tutorial.

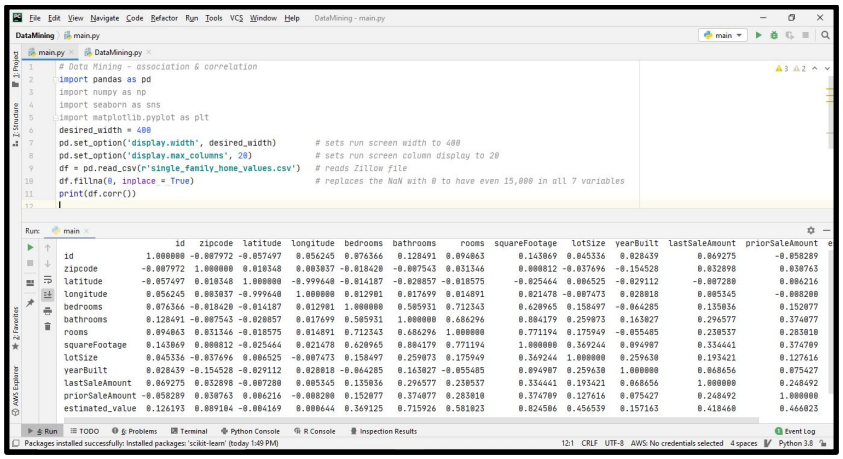
Association & Correlation:

Association is measured by coefficients of correlation or determination. As a measure of association, the coefficient of correlation (i.e., r) is part of a covariance matrix, where correlation measures if a relationship between two variables is direct (i.e., positive) or inverse (i.e., negative). A coefficient of correlation falls between -1 <= r <= 1. A measure close to zero reflects no association, a measure near -1 reflects an inverse relationship, and a measure near 1 reflects a direct relationship. Coefficient of determination (i.e., r^2) reflects the strength of association between variables that explain variance, where near 0 (i.e., none) reflects little explanation while near to 1 (i.e., strong) reflects stronger explanation. Correlation is part of the covariance matrix, dispersion matrix, variance matrix, or variance-covariance matrix. It is a matrix in which i-j position defines the correlation between the ith and jth parameter of the given data-set.

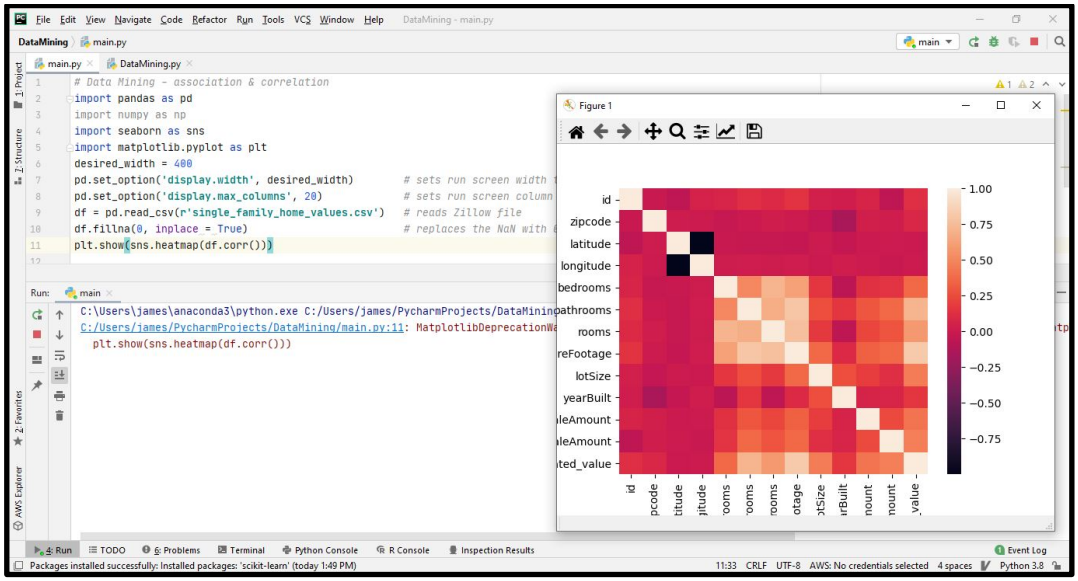
When the data points follow a roughly straight-line trend, the variables are said to have an approximately linear or direct relationship. In some cases, the data points fall close to a straight line, but more often there is quite a bit of variability of the points around the straight-line trend. Correlation describes the strength of the linear association. Correlation summarizes the strength and direction of the linear (straight-line) association between two quantitative variables. Denoted by r, it takes values between -1 and +1. A positive value for r indicates a positive association, and a negative value for r indicates a negative association.

The closer r is to 1 the closer the data points fall to a straight line, thus, the linear association is stronger. The closer r is to 0, making the linear association weaker.

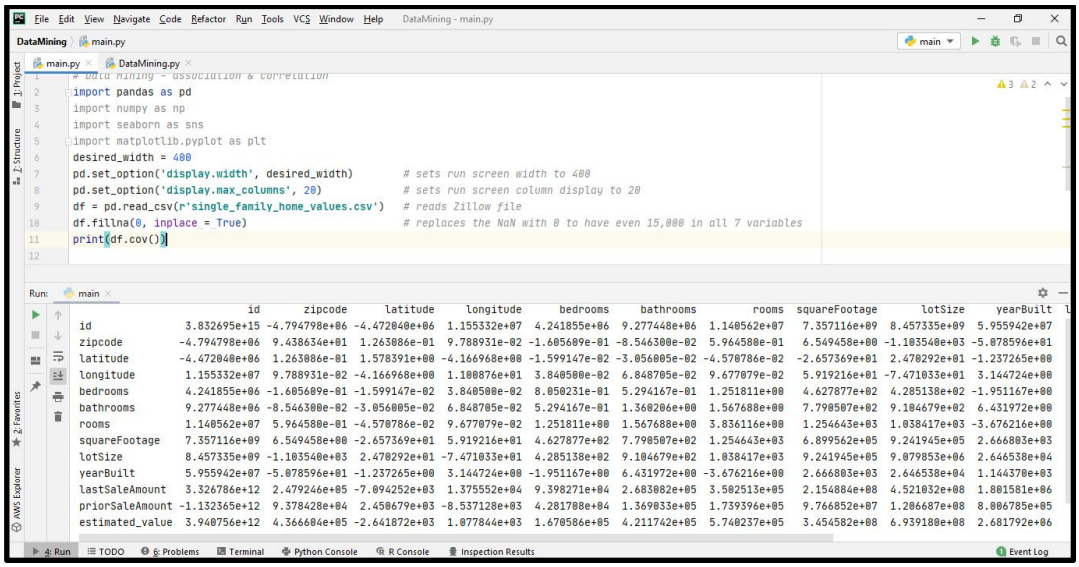
1) The Python code using the df.corr() function in the PyCharm screenshot below, provides a simple correlation matrix between all non-string column attributes:



2) For more visualization, the PyCharm screenshot below depicts a heat map of correlation between variables and the darker squares indicate higher correlations between variables. Note the diagonal of the matrix is white because it is the correlation of a variable to itself.

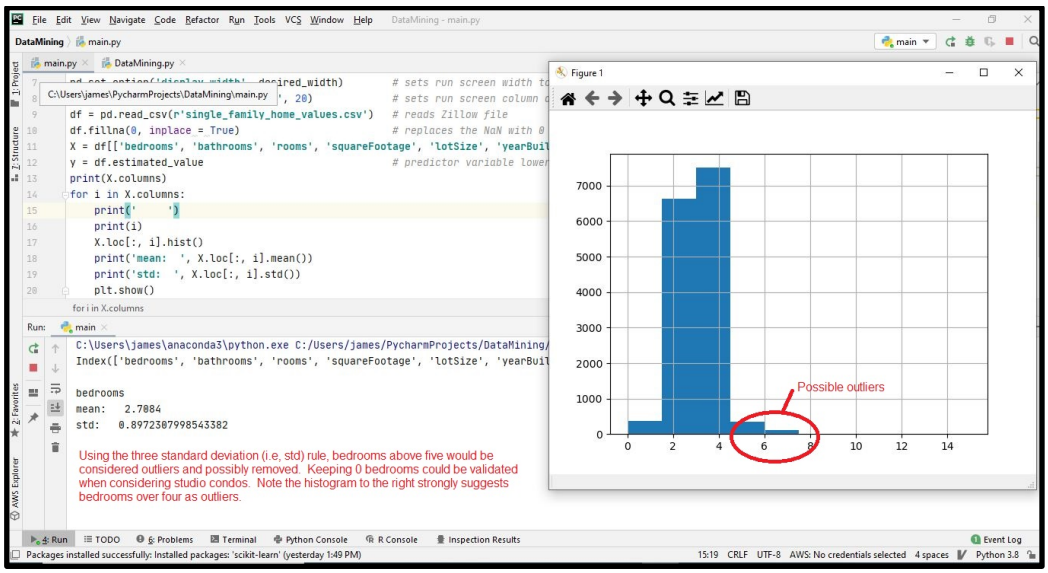


3) Covariance is the unstandardized correlation and can be identified in a matrix by the Python code dataframe.cov() as illustrated in the PyCharm screenshot below.

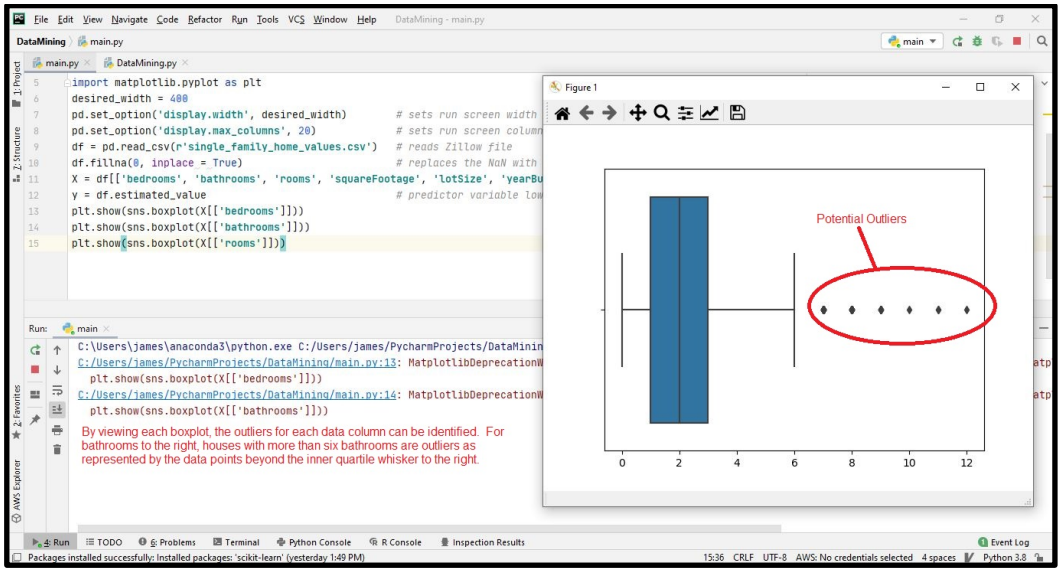


4) Measures of association help data scientists identify which variables explain other variables in terms of variance. Coefficient of correlation, covariance, and a coefficient of determination are all calculated in terms of variance. Outliers generate noise by artificially amplifying variance. Beyond correlation and covariance, two measures to identify outliers are the three standard deviation rule and boxplots identifying IQRs. Using the three standard deviation rule, we would exclude all outliers that are three standard deviations from the mean. The PyCharm

screenshot below uses a 'for loop' to display histograms for each of our seven columns (e.g., bedrooms, bathrooms, rooms, squareFootage, lotSize, yearBuilt, and priorSaleAmount). For bedrooms, the three standard deviation rule suggests 2.784 +/- (3)(0.892307998543382). Per the justification in the PyCharm screenshot below, a case can be made to keep data points of houses having between 0 to 4 and 0 to 5 bedrooms.



Another tool to use in outlier identification is a boxplot that uses inter-quartile ranges IRQs. The following PyCharm screenshot shows the Python code to produce boxplots for the X column data points.



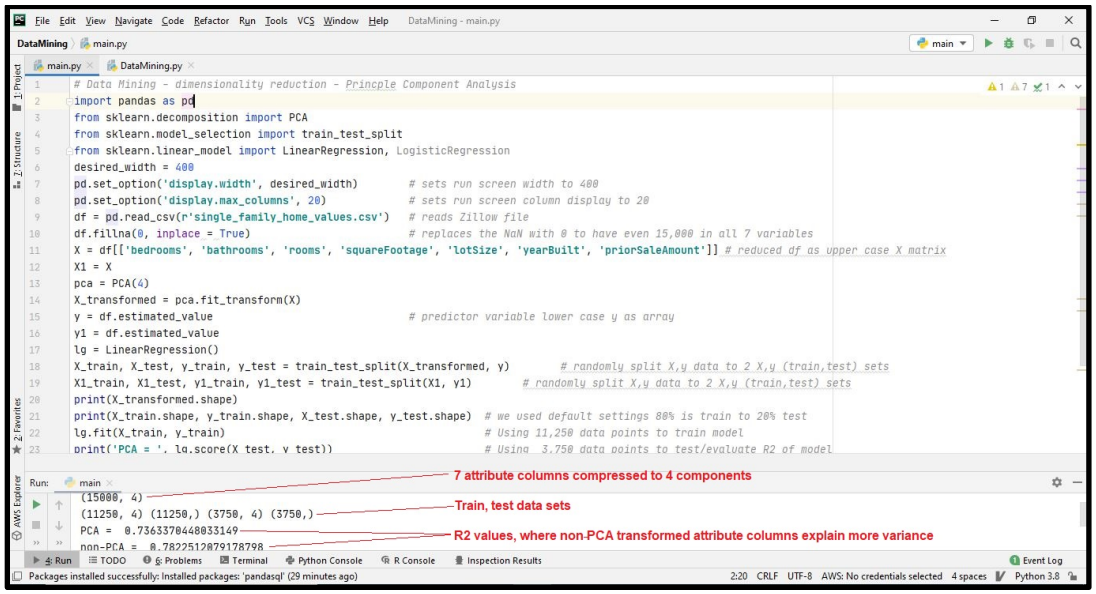
5) **Exercise deliverable:** (1)Construct an example of df.corr() function or df.cov() in Python code using the matrix or a heat map of the matrix to display the correlation or covariance between column attributes. (2) As a second PyCharm screenshot, use the three standard deviation rule

or a boxplot to identify outliers. Use either the single_family_home_values.csv or your data project csv. In using the CSV file above, choose different column attributes than those shown in the previous two screenshots. Save the two PyCharm screenshots for use later in this tutorial.

Dimensionality Reduction:

Principle Component Analysis (PCA): PCA is used to decompose or compress a multivariate dataset into a set of successive orthogonal components that explain a maximum amount of the variance. In sklearn, PCA is implemented as a transformer object that learns components in its fit method, and can be used on new data to project it on these components. The focus of PCA is to compress the number of explanatory variables by taking portions of starting variables and compressing them into a given number of components. For more details on PCA, refer to the following URL: https://scikit-learn.org/stable/modules/decomposition.html#pca

1) The following PyCharm screenshot demonstrates how PCA can condense seven attribute columns into four components. The R^2 of the PCA compressed components show how explained variance can be lost when compressing explanatory variables into fewer components, especially when the reduction is small (i.e., 7 to 4).



2) **Exercise deliverable:** Use PCA to reduce the number of explanatory variables into fewer components. Use either the single_family_home_values.csv or your data project csv. If you choose the csv file above, do not use the exact same attributes. Save the PyCharm screenshot for use later in this tutorial.

Deliverables from Exercises Above:

Take the noted deliverables (i.e., PyCharm screenshots) from each exercise above (i.e., total of 8 screenshots), insert the screenshots into a MS Word document as you label each screenshot deliverable from the corresponding exercise, save the document as

one PDF, and submit the PDF to the Tutorial Eight-A Canvas Assignment uplink having the following qualities:

a. Your screenshots for each of the four exercise deliverables above is included.
b. Each screenshot is labeled appropriately for each exercise.
    i. Cluster Analysis …….. 2 screenshots
    ii. Regression and Classification ……. 3 screenshots
    iii. Association and Correlation ……. 2 screenshots
    iv. Dimensionality Reduction ……. 1 screenshot
    Total PyCharm screenshots to submit …… 8 screenshots
c. All screenshots are legible output for each PyCharm exercise deliverable.