**Module 8**

* [Video Transcripts](https://student.emeritus.org/courses/4765/files/3118787?wrap=1)
* [Download Video Transcripts](https://student.emeritus.org/courses/4765/files/3118787/download?download_frd=1)
* [Quick Reference Guide](https://student.emeritus.org/courses/4765/files/3118796?wrap=1)

Had to upgrade scikit-learn package

% conda config --set ssl\_verify no

% conda install scikit-learn

% conda config --set ssl\_verify yes

% conda list | grep scikit

**Matilde’s Session**

If correlation of variables is more than 70%, eliminate variables - R-squared covariance

**Savio’s Session**

VIF: Variance inflation Factor to determine Multicollinearity in dataset

Permutation importance similar to vif!

* Lower the drop in r^2 lesser the importance!
* LinearRegression().fit(X,y)
* lr\_model.score(X,y) is R^2

# vif

def vif(exogs, data):

vif\_dict = {}

for exog in exogs:

not\_exog = [i for i in exogs if i!=exog]

X,y = data[not\_exog], data[exog]

# vif = 1 / (1- R^2)

r\_squared = LinearRegression().fit(X,y).score(X,y)

vif = 1 / (1-r\_squared)

vif\_dict[exog] = vif

df\_vif = pd.DataFrame({"VIF":vif\_dict})

return df\_vif

vif(X.columns,X).sort\_values(by = "VIF", ascending = False)

**Notes:**

*inference* is generally used to conclude something about the present, while *prediction* always pertains to a future event or occurrence.

In machine learning, the word variance is often used to represent the sensitivity of a model to the training data.

The **permutation feature importance** is defined to be the decrease in a model score when a single feature value is randomly shuffled [1](https://scikit-learn.org/stable/modules/permutation_importance.html#id2).

A training dataset is the initial data used to train a machine learning model. The training dataset assists the model in learning how to apply concepts in order to produce results. This includes describing input data as well as describing the expected output. Training data may be accompanied by subsequent datasets called validation and testing sets.

Once the fitted model has been determined, the observations in the second set of data, called the validation set, are predicted using the fitted model. The validation set is used to evaluate whether a model fits a training set without bias.

Last but not least, the test dataset provides an unbiased evaluation of the final model fit on the training dataset. If the test dataset has not been used in training (such as in cross-validation), the test dataset is called a holdout dataset. In addition, when the original dataset is divided into only two subsets, the test set may be referred to as the validation set.

In conclusion, the steps to most optimally train your dataset are as follows:

1. Divide your data into training, validation, and test sets. Ultimately, the rule of thumb is that these should be divided into 60/20/20 splits.
2. Choose a training algorithm and set of parameters
3. Train the model based on the training set
4. Evaluate the model based on the validation set
5. Repeat steps two through four with different training parameters and algorithms
6. Choose the best model and train it using the data from the training and validation sets
7. Evaluate this final model using the test data

**Plotting MSEs:**

print(f'The Complexity that minimized Test Error was: {test\_mses.index(min(test\_mses)) + 1}')

plt.plot(range(1, 21), train\_mses, '--o', label = 'training error')

plt.plot(range(1, 21), test\_mses, '--o', label = 'testing error')

plt.xticks(range(1, 21), range(1, 21))

plt.xlabel('Degree Complexity')

plt.ylabel('Mean Squared Error')

plt.legend()

**Module Issues:**

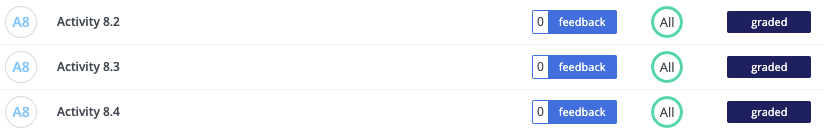
Site-wide issue: error 526 invalid ssl certificate on May 4th.

Codio Activity 8.2 Problem 4 variable name is supposed to be two\_feature\_poly\_df.

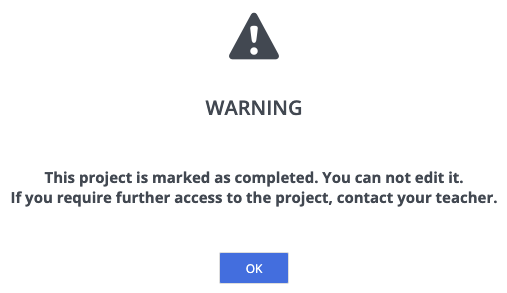
Codio Activity 8.6 Problem 2 random\_state is supposed to be 32.

Codio Activity 8.7 The Data: Ames Housing has a broken link: <https://lilyohio-sonicjaguar-3000.codio.io/notebooks/data/data_description.txt>

The grading failed on Codio Activity 8.2, 8.3, 8.4. I cannot even "reset" this activity because this option is not available on this exercise, the message stating "The activity marked as completed".



The grading failed on Codio Activity 8.4, no feedback was provided. I was able to reset the activity, retired it but failed to grade again with no feedback.

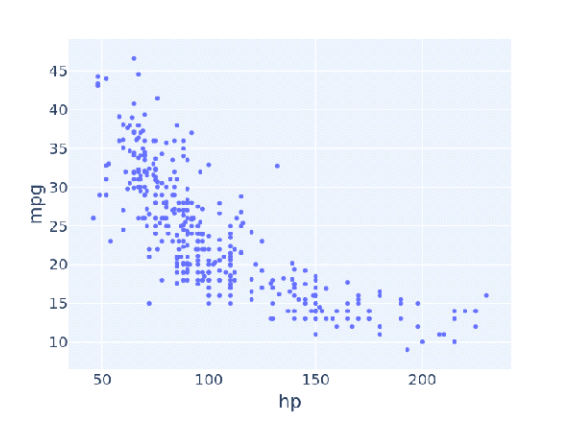


**Quizes:**

If there are three features, the linear regression model predicts an output y^ = (blank). : *θ1 Φ1* + *θ2 Φ2* + *θ3 Φ3*

*You are correct! The answer “θ1 Φ1* + *θ2 Φ2* + *θ3 Φ3” is correct because this is the formula for a three-feature linear regression model.*

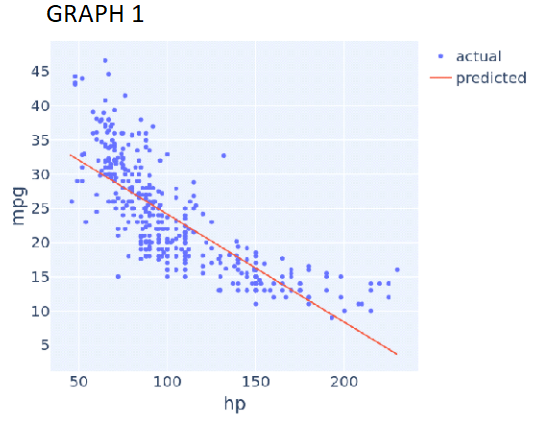
The graph shows that as horsepower or ‘hp’ increases the miles per gallon or ‘mpg’ increases. : False

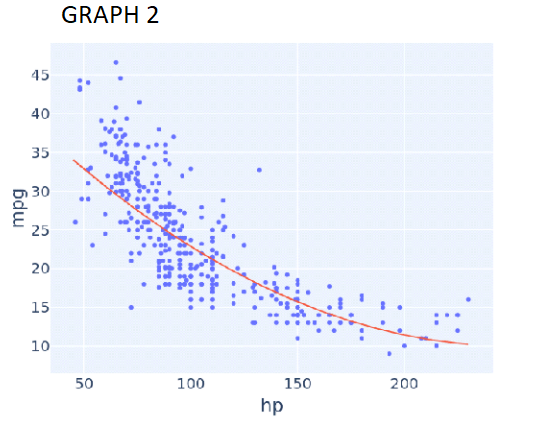


*You are correct! The answer “*False*” is correct because the graph shows that with the increase in horsepower or ‘hp’, the miles per gallon or ‘mph’ decreases.*

Which of the following graphs is an example of a parabola approach? : GRAPH 2

*You are correct! The answer “*GRAPH 2*” is correct because the trend line in the graph is curved.*





A simple linear regression model can have squared terms. : False

*You are correct! The answer “*False*” is correct because a simple linear regression model cannot have squared terms.*

How do you include squared terms in a linear regression model? : Adding a new column with squared values of the feature

*You are correct! The answer “*Adding a new column with squared values of the feature*” is correct because to include squared terms in a linear regression model, the solution is to add a new squared column of that feature.*

The library in Python “from sklearn.linear\_model import Linear regression” can be used for squared regression model building. : True

*You are correct! The answer “*True*” is correct because it can be used for a squared regression model introducing squared terms of features.*

A transformer takes a set of existing features as input and outputs new features. : True

*You are correct! The answer “True” is correct because the transformer takes a set of existing features as input and outputs new features.*

In PolynomialFeatures(degree), ‘degree’ is used to declare the square size. : False

*You are correct! The answer “*False*” is correct because the constructor ‘degree’ of the polynomial is used to control the number of features added.*

A PolynomialFeatures object with the ‘degree’ constructor equal to ‘5’ is written as (blank). : PolynomialFeatures(degree=5)

*You are correct! The answer “*PolynomialFeatures(degree=5)*” is correct because this is the syntax in Python to create a “PolynomialFeatures” object with the “degree constructor equal to 5”.*

“PolynomialFeatures().fit\_transform()” will output a (blank). : Numpy array

*You are correct! The answer “*Numpy array*” is correct because the output of “*PolynomialFeatures().fit\_transform()*” is a “Numpy array”.*

What is the output of the function “.get\_feature\_names\_out()” for the “PolynomialFeatures” object, given the feature name as “hp” and the degree as ‘3’? : [“hp”,”hp^2”,”hp^3”] ->  [“hp”,”hp2”,”hp3”]

*You are correct! The answer “*[“hp”,”hp2”,”hp3”]*” is correct because this is the output of the function to get the feature names.*

[“hp”,”hp2”,”hp3”]

Consider the following linear regression pipeline model with a polynomial transform degree equal to 3:

**vehicle\_data\_with\_cubic\_features =               pd.DataFrame(poly\_transform.fit\_transform(vehicle\_data[["hp"]]),              columns = poly\_transform.get\_feature\_names\_out()) cu\_model.fit(vehicle\_data\_with\_cubic\_features, vehicle\_data[["mpg"]])**

The Python statement for the prediction of a 100-horsepower vehicle would be cu\_model.predict([[100]]). : False

*You are correct! The answer “*False*” is correct because the Python statement for the prediction of a 100-horsepower vehicle would be*

cu\_model.predict([[100,10000,1000000]]).

Consider the following linear regression pipeline model with a polynomial transform degree equal to 3:

**vehicle\_data\_with\_cubic\_features =** **pd.DataFrame(poly\_transform.fit\_transform(vehicle\_data[["hp"]]),** **columns = poly\_transform.get\_feature\_names\_out())** **cu\_model.fit(vehicle\_data\_with\_cubic\_features, vehicle\_data[["mpg"]])**

What would the Python statement for the prediction of a 100-horsepower vehicle be? : cu\_model.predict([[100,10000,1000000]]), cu\_model.predict(poly\_transform.fit\_transform[[100]])

*You are correct! The answer “*cu\_model.predict([[100,10000,1000000]])*" is correct because it is a correct Python statement for the prediction of a 100-horsepower vehicle.*

*The answer “*cu\_model.predict(poly\_transform.fit\_transform[[100]])*”  is correct because it is a correct Python statement for the prediction of a 100-horsepower vehicle.*

Scikit-learn’s pipeline class is a useful tool for encapsulating multiple different transformers alongside an estimator into one object. : True

Yo*u are correct! The answer “*True*” is correct because pipeline class is a useful tool for encapsulating multiple different transformers alongside an estimator into one object.*

Consider the following linear regression pipeline model:

**from sklearn.pipeline import Pipeline** **pipelined\_model = Pipeline([** **('josh\_transform', PolynomialFeatures(degree = 3, include\_bias = false)),** **('josh\_regression', LinearRegression())** **])** **pipelines\_model.fit(vehicle\_data[["hp"]], vehicle\_data["mpg"])**

The Python statement for the prediction of a 100-horsepower vehicle would be Pipelined\_model.predict([[100]]). : True

*You are correct! The answer “*True*” is correct because this is the correct statement used for the prediction of a pipelined model.*

Consider the following pipelined regression model:

**from sklearn.pipeline import Pipeline** **pipelined\_model = Pipeline([** **('josh\_transform', PolynomialFeatures(degree =  , include\_bias = False)),** **('josh\_transform', LinearRegression())** **])** **pipelined\_model.fit(vehicle\_data[["hp"]], vehicle\_data["mpg"])**

For a model to be of order six, what should be the value (highlighted) of the constructor degree? : 6

*You are correct! The answer “*6*” is correct because the constructor should be set to value “6” for a model to be of order six.*

Increasing model complexity seems to decrease the error. : True

*You are correct! The answer “*True*” is correct because increasing the model’s complexity seems to decrease the error.*

Increasing model complexity seems to decrease the variance. : False

*You are correct! The answer “*False*” is correct because increasing the model’s complexity seems to increase the variance.*

Being overly sensitive to data is known as overfitting. : True

*You are correct! The answer “*True*” is correct because being overly sensitive to data is known as overfitting.*

If there are N data points, then a degree N-1 model will also have an MSE equal to 1. : False

*You are correct! The answer “*False*” is correct because if there are N data points, then a degree N-1 model will also have an MSE equal to 0.*

If there are four data points, there can always be a four-parameter model, three coefficients, and one (blank) that fit perfectly. : Intercept (alpha)

*You are correct! The answer “*Intercept*” is correct because in a four-parameter model, there are always three coefficients and one intercept that fit perfectly.*

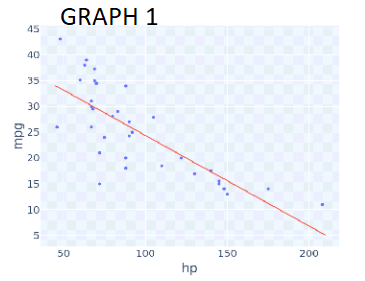
Suppose that the number of numerical features available exceeds the number of data points. If a linear regression model is used, there will be zero training error. : True

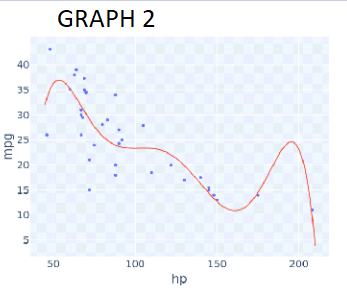
*You are correct! The answer “*True*” is correct because when the number of numerical features available exceeds the number of data points in a linear regression model, there will be zero training error.*

The cross-validation technique is used to detect the degree of the model build. : False

*You are correct! The answer “*False*” is correct because the cross-validation technique is used to detect the overfitting of the model build.*

Which of the following graphs seems to be overfitting? : GRAPH 2





*You are correct! The answer “*GRAPH 2*” is correct because it can be seen that the curve of the graph is passing through a lot of data points and has a wiggly shape.*

In simple cross-validation the data is divided into which two sets? : Training set, Development set

*You are correct! The answer “*Training set*” is correct because in simple cross-validation, it is one of the two datasets into which this data can be divided.*

*You are correct! The answer “*Development set*” is correct because in simple cross-validation, it is one of the two datasets into which this data can be divided.*

The development set is used to compare models before they are trained. : False

*You are correct! The answer “*False*” is correct because the development set is used to compare models after they are already trained.*

Consider the following code:

**from sklearn.utils import shuffle** **training\_set, dev\_set = np.split(shuffle(vehicle\_data\_sample\_35), [25])**

Given this code, the variable training\_set will have \_\_ rows if the total data size is 35. : 25

*You are correct! The answer “*25*” is correct because in the code, the initial division point defined is [25].*

What is the common shape of the validation error curve? : Bowl shaped

*You are correct! The answer “*Bowl shaped*” is correct because the shape of the validation error curve is mostly bowl shaped.*

Hyperparameters are parameters whose values control the learning process. : True

*You are correct! The answer “*True*” is correct because hyperparameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning.*

The training set obtained from the split of the datasets is used for the selection of hyperparameters. : False

*You are correct! The answer “*False*” is correct because training sets are used to train models of various complexity.*

In machine learning, the final estimate of outcome is ideally done on a third dataset. What is this third dataset commonly called? : Test set

*You are correct! The answer “*Test set*” is correct because in machine learning, the final estimate of outcome is done on test sets.*

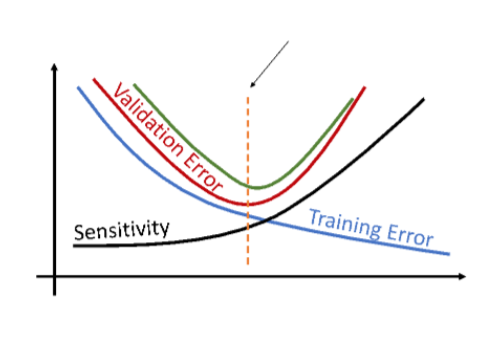
Consider the following Python code:

**diamond\_training\_data, diamond\_validation\_data, diamond\_test\_data = np.split(diamond\_data, [1500, 1800])**

For the datasets of 2000 rows, what would be the size of the Diamond\_test\_data? : 200

*You are correct! The answer “*200*” is correct because in the given code, the initial division point is 1500 and the second is 1800; therefore, the remaining 200 rows out of 2000 will be the size of*diamond\_test\_data*.*

In the given graph, the green curve represents development error. : False



*You are correct! The answer “*False*” is correct because* *the green curve in the graph represents test error.*

**Discussion 8.1: Prediction vs. Inference - Section B**

A table outlining the differences between "inference" and "prediction".

|  |  |  |
| --- | --- | --- |
| **Parameters for Comparison** | **Inference** | **Prediction** |
| Definition | Inference describes a conclusion that is based on empirical evidence regarding data, facts, and evidence. | Prediction describes a conclusive statement that pertains to a future event or occurrence. |
| Answers the question | What do the relationships between the variables mean? | How can I accurately predict new data points? |
| Goal | Estimate an association between an outcome, variable, and a predictor variable. | Develop a ‘best’ model (where all predictors predict Y with high accuracy and low error). |
| Example | What has the biggest impact on fuel efficiency: horsepower or weight? | Given horsepower, what is a vehicle’s fuel efficiency? |
| Confidence of conclusion | Since inference is based on facts and evidence, you can have more confidence that the conclusion is correct. | Compared to inference, there is less certainty that a prediction will be correct because the future is unknown. |

Autonomous cars are inference model, their conclusion should be correct all the time to avoid collusions and fatality.

**Planing for a road trip**

When planning for a road trip between two points, I can infer how long it may take by considering the distance.

Why would your inference and prediction be correct?

It is mostly straight forward calculation because trip duration is highway mileage divided by average hourly speed which are known.

What are the weaknesses of your inference and prediction?

There are several unknowns like traffic, accidents and weather related delays along the road.

What data would make it a more precise inference or prediction?

Knowing weather forecast, current traffic conditions and accidents may help come up with alternative routes

What would be the dangers of decision-making based on your inference and prediction?

Getting stuck in traffic may lead to running out of gas, or strand or night time driving just to name a few potential issues where drivers and passengers must be prepared.

**Try-It Activity 8.1: The “Best” Model - Section B**

# permutation importance

from sklearn.inspection import permutation\_importance

r = permutation\_importance(lr\_model1, X1\_test, y\_test, n\_repeats = 10, random\_state = 123)

First dataset analysis and preparation:

Try up to 5 degrees:

# Check best MSEs among all degrees

train\_mses = []

test\_mses = []

for d in range(1,6):

#create pipeline with PolynomialFeatures and LinearRegression

#remember to set include\_bias = False

#pfeatures = PolynomialFeatures(degree = d, include\_bias = False)

#model = pfeatures.fit\_transform(X[['total\_rooms']])

pipe = Pipeline([('features', PolynomialFeatures(degree = d, include\_bias = False)), ('model', LinearRegression())])

#fit on training data

pipe.fit(X\_train, y\_train)

#mse of training data

train\_mse = mean\_squared\_error(y\_train, pipe.predict(X\_train))

train\_mses.append(train\_mse)

#mse of testing data

test\_mse = mean\_squared\_error(y\_test, pipe.predict(X\_test))

test\_mses.append(test\_mse)

best\_model\_complexity = test\_mses.index(min(test\_mses)) + 1

print(type(best\_model\_complexity))

print(best\_model\_complexity)

MSEs:

All:

[4890.524091257272,

16697180982819.168,

13730260.365321374,

2563265.538472204,

53476957180.40422]

[5299.116072934263,

78103515637541.23,

9750479.038576635,

116534.93578107428,

7278.368046118499]

# plot

print(f'The Complexity that minimized Test Error was: {test\_mses.index(min(test\_mses)) + 1}')

plt.plot(range(1, 6), train\_mses, '--o', label = 'training error')

plt.plot(range(1, 6), test\_mses, '--o', label = 'testing error')

plt.xticks(range(1, 6), range(1, 6))

plt.xlabel('Degree Complexity')

plt.ylabel('Mean Squared Error')

plt.legend()



**Dataset Analysis and Cleanup**

Transform ocean\_proximity:

ca=pd.get\_dummies(cali, prefix=['ocean\_proximity'])

# fill in missing total\_bedrooms by mean ratio of total\_bedrooms to total\_rooms

# those rows can also be simply dropped b/c affecting only 1% of data

ratio = ca['total\_bedrooms'].mean()/ca['total\_rooms'].mean()

ca['total\_bedrooms'] = ca['total\_bedrooms'].fillna(ratio \* ca['total\_rooms'])

# Minimize footprint on median\_house\_value for calculating higher degrees of model

ca['median\_house\_value'] = ca['median\_house\_value']/1000

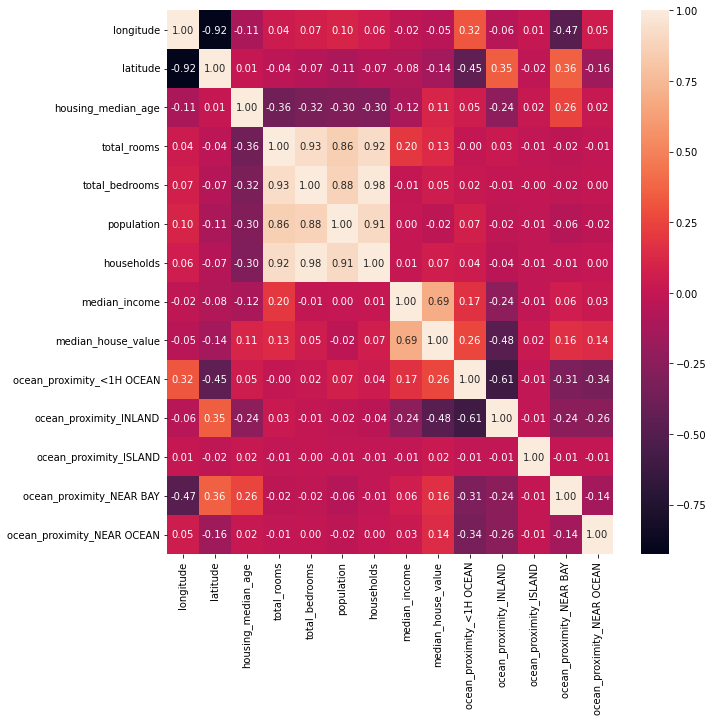
Check correlated columns:

#heatmap

plt.tight\_layout()

plt.subplots(figsize=(10,10))

sns.heatmap(ca.corr(), annot=True, fmt='.2f')



There are strong positive correlation among those 4 fields, I will attempt to remove them by checking the model R^2 score.

**Prepare dataset**

# Prepare dataset

Xall = ca[['longitude', 'latitude', 'housing\_median\_age', 'total\_rooms',

'total\_bedrooms', 'population', 'households', 'median\_income',

'ocean\_proximity\_<1H OCEAN',

'ocean\_proximity\_INLAND', 'ocean\_proximity\_ISLAND',

'ocean\_proximity\_NEAR BAY', 'ocean\_proximity\_NEAR OCEAN']]

y = ca['median\_house\_value']

# Split dataset

Xall\_train, Xall\_test, y\_train, y\_test = train\_test\_split(Xall, y, random\_state = 32, train\_size=0.7, test\_size=0.3)

Assess best performing degree

# Check best MSEs among all degrees

train\_mses = []

test\_mses = []

for d in range(1,6):

#create pipeline with PolynomialFeatures and LinearRegression

pipe = Pipeline([('features', PolynomialFeatures(degree = d, include\_bias = False)), ('model', LinearRegression())])

#fit on training data

pipe.fit(Xall\_train, y\_train)

#mse of training data

train\_mse = mean\_squared\_error(y\_train, pipe.predict(Xall\_train))

train\_mses.append(train\_mse)

#mse of testing data

test\_mse = mean\_squared\_error(y\_test, pipe.predict(Xall\_test))

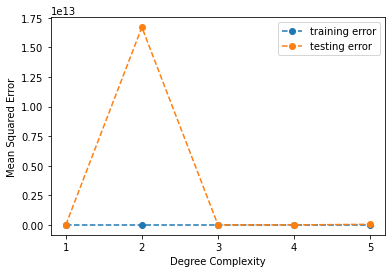
test\_mses.append(test\_mse)

best\_model\_complexity = test\_mses.index(min(test\_mses)) + 1

print(type(best\_model\_complexity))

print(best\_model\_complexity)

I executed LinearRegression() with higher degrees up to 5, the best MSE was with degree=1 of those 5 runs:



MSE Values per run:

[4890.524091257272,

16697180982819.168,

13730260.365321374,

2563265.538472204,

53476957180.40422]

First, I executed with all columns to get r-square and feature importance

# fit model with training set

model = LinearRegression().fit(Xall\_train, y\_train)

# score with test set

model.score(Xall\_test, y\_test)

r^2 : 0.6334205640890778

# permutation importance

r = permutation\_importance(model, Xall\_test, y\_test, n\_repeats = 10, random\_state = 32)

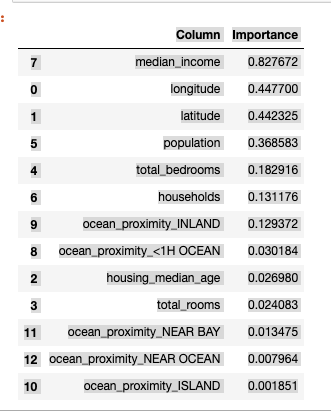
r.importances\_mean

array([0.44770039, 0.44232519, 0.0269804 , 0.02408259, 0.18291621,

0.36858329, 0.13117628, 0.82767194, 0.03018423, 0.12937226,

0.00185126, 0.01347451, 0.00796388])

pd.DataFrame({"Column":Xall.columns, "Importance":r.importances\_mean}).sort\_values(by = "Importance", ascending = False)



Created a function to try out best r-squared per column:

#function to execute permutation importance!

def column\_importance(X, y):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state = 32, train\_size=0.7, test\_size=0.3)

# fit model with training set

model = LinearRegression().fit(X\_train, y\_train)

# score with test set

print('model r^2 :', model.score(X\_test, y\_test))

# permutation importance

r = permutation\_importance(model, X\_test, y\_test, n\_repeats = 10, random\_state = 32)

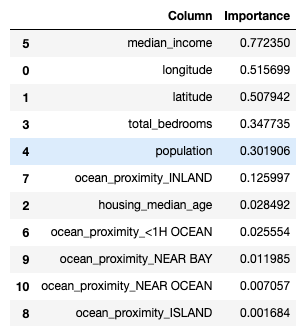
print('importance:', r.importances\_mean)

return model, X\_test, y\_test, pd.DataFrame({"Column":X.columns, "Importance":r.importances\_mean}).sort\_values(

by = "Importance", ascending = False)

I found best r-squared with this combination:

model r^2 : 0.6342275269087095 which is slightly better than the initial set: 0.6334205640890778!



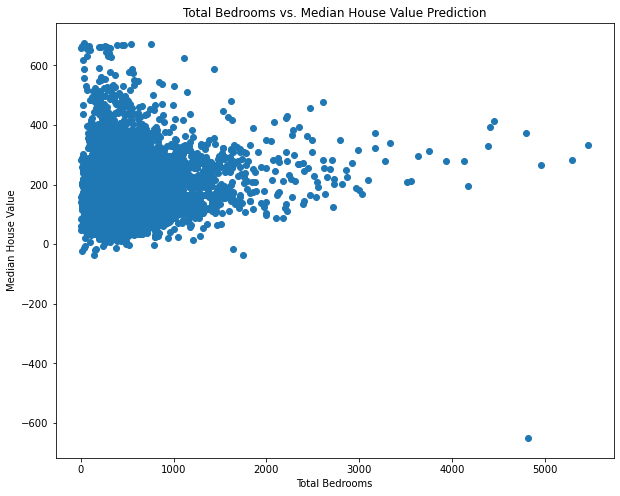
Where I removed total\_rooms and households. Although, population correlated to total\_bedrooms, removing it has negatively affecting r^2, similarly, any low importance column above. So, I left them in the dataset.

I also checked visual performance of model, the quality is not good because it predicts negative median\_house\_value!

plt.tight\_layout()

plt.subplots(figsize=(10,8))

plt.scatter(x=X\_test['total\_bedrooms'], y = model.predict(X\_test))



Y dependent variable, X independent variables