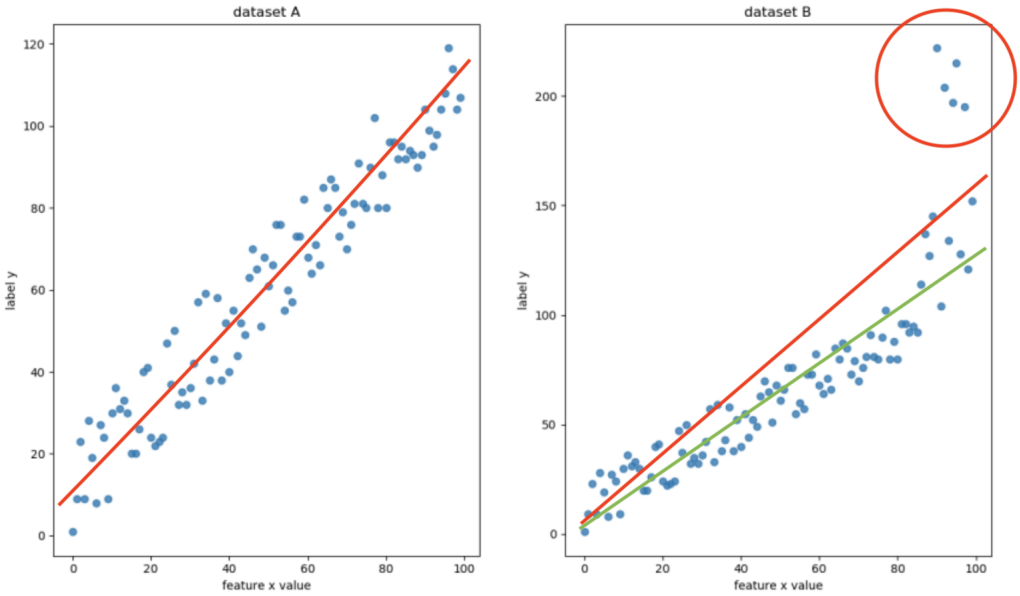
1) - 4) turned in in modules 2.1 and 2.2.

5)
a) Red line on graph A.
b) Red line on graph B.
c) Circle on graph B.
d) Green line on graph B.
e) Removing the possible outliers would create a smaller mean squared error because the outlier points have a big effect on the (Predicted_i - Actual_i) terms.



6)
a) Another way we can frame the problem is we want to predict the median house age in districts in California. This is a supervised learning problem since we have the response labels. This is a regression problem since response is numerical.

b) Based on the correlation matrix, we can conclude the coefficient sign for median_income(B7), total_rooms(B3), housing_median_age(B2), households(B6), total_bedrooms(B4) are positive. These values in the correlation matrix are positive. We can conclude the coefficient sign for population(B5), longitude(B0), latitude(B1) are negative. These values in the correlation matrix are negative. The correlation matrix represents the magnitude of the relationship between variables, and that is similar to the interpretation of the coefficients.

c) Looking at the describe function and plots, I decided to use 6 groups for housing_median_age. The data was spread between 0 - 50 years mostly, but has a spike at 51-53, so I included a 6th category.

```
housing["age_cat"] = pd.cut(housing["housing_median_age"],
                            # TODO :: split the feature into different categories, expect 1 line of code
                            bins=[0., 10, 20, 30, 40, 50, np.inf],
                            # TODO :: label for each category, expect 1 line of code)
                            labels=[1, 2, 3, 4, 5, 6])

# TODO :: use StratifiedShuffleSplit to split the dataset into training set and test set while preserving
# the percentage of samples for each category.
# expect 4 lines of code
split2 = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split2.split(housing, housing["age_cat"]):
    strat_train_set = housing.loc[train_index]
    strat_test_set = housing.loc[test_index]
```

d) One hot encoding could be use when the category doesn't have a huge amount of different values and the data is not ordinal. Therefore, it could be used for a 'closest_city' feature.

e) Having 0 error on the training data set probably means something went wrong or the model severely overfits the data. The error for test data would probably be very high for the model. It's usually always possible to increase model complexity to generate a lower error on the training data, but in machine learning, we care about how well the model predicts new data that it has not been seen.

f)
The best SVR predictor is linear and gets a RMSE of 105773.779. kernel rmse linear: 105773.779 rbf: 117550.993 poly: 117330.949 sigmoid: 117101.023 precomputed: can't use, must be square

g)
GridSearchCV
final_rmse: 49491.61792572629 [47312.03285458, 51579.18220542]

RandomizedSearchCV
final_rmse: 50560.678 [48372.88497456, 52657.65329852]