HOMEWORK ASSIGNMENT – 4

ANSHIKA SHARMA, UCLA ID: 305488635

PART -I

In this part, I used the feedforward neural network (FNN) model on the California housing dataset (regression). The dataset includes 20640 samples and each sample is a California district. The target variable is the median house value for California districts.

As a first step, I imported packages and then set seed. Then I split the data randomly, 20640 samples (and their labels) into two halves for training models and testing purposes, respectively. Then, I randomly split the 10320 training samples into two subsets: a subset of 2064 samples for validation purpose and the rest 8256 samples for training. I used the third-party function, "train_test_split" for this splitting operation.

For the second step, We need to carry out model selection and are required to specify at least three sets of hyper-parameters which include the number of hidden layers, learning rate and activation function. MSE is used as a loss function instead of MAE because it makes the estimation precise. I created six models with different learning rates, different activation functions and different number of hidden layers. For each, I used tensorflow to train the FNN model on the training samples, and then apply the learned model over the validation set (8256 samples). For each model I found the results and computed R^2 (determination of coefficients).

	R2_score	Hidden_layer	activation_function	Learning_rate
mod_1	-0.000122335	2	relu,relu	0.0100
mod_2	-0.00685839	2	relu, sigmoid	0.0100
mod_3	-1.78934e-06	2	tanh, softmax	0.0100
mod_4	0.476327	2	relu,relu	0.0001
mod_5	0.519603	3	relu,relu,relu	0.0010
mod_6	0.626026	3	relu,relu,sigmoid	0.0010

The model that achieves the top R^2 is mod_6 with R^2 of 0.626026 or 62.60%. This model has 3 hidden layers, a learning rate of 0.0010 and the activation function used in each layer respectively is "relu", "relu" and "sigmoid".

The top ranked model based on R^2 is mod_6. As a part of the third step,
I apply mod_6 over the testing samples (10320 samples). The R^2 over testing samples is as follows:

	R2_score
mod_6	0.631266

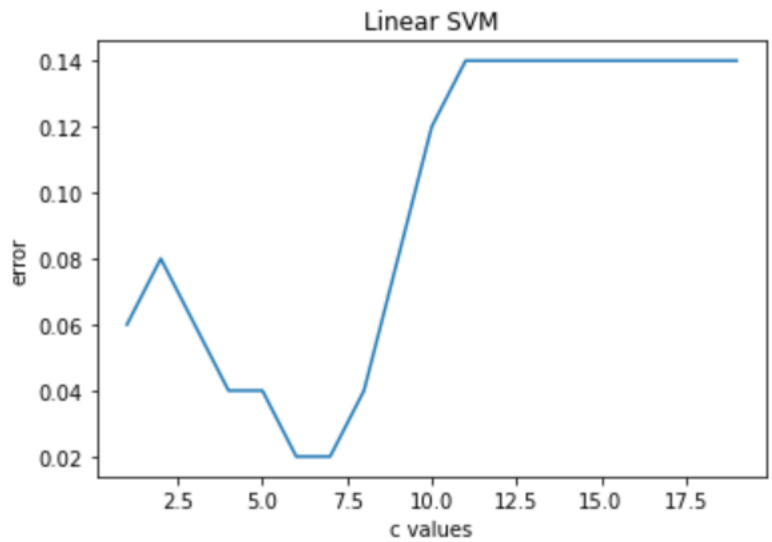
In step 4, I analyze the testing results of my top-ranked model, mod_6. For each testing sample, I calculated the absolute error between the model's prediction and ground-truth value (both are real-valued). Then, as can be seen in the table I have reported ten testing samples which received the largest absolute errors.

	Y	Y_hat	abs(error)	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
1902	5.000010	1.193820	3.806190	0.499900	29.000000	2.373272	1.055300	2690.000000	12.396313	34.020000	-118.280000
7378	5.000010	1.193822	3.806188	1.169600	52.000000	2.436000	0.944000	1349.000000	5.396000	37.870000	-122.250000
2781	5.000010	1.193838	3.806172	0.702500	19.000000	2.425197	1.125984	1799.000000	2.833071	35.300000	-120.670000
5291	5.000010	1.193840	3.806170	4.203900	11.000000	6.753927	1.031414	881.000000	4.612565	37.190000	-121.740000
8325	5.000010	1.297806	3.702204	0.854300	27.000000	2.297872	1.175532	1211.000000	1.610372	37.780000	-122.420000
4604	1.125000	4.814892	3.689892	12.538100	29.000000	6.888889	1.222222	50.000000	2.777778	33.960000	-117.440000
8206	5.000010	1.324971	3.675039	5.206600	4.000000	10.500000	1.445652	311.000000	3.380435	33.510000	-117.320000
4739	5.000000	1.357489	3.642511	2.353600	26.000000	2.826563	1.000000	2543.000000	3.973438	34.050000	-118.310000
6857	5.000010	1.359542	3.640468	4.975700	35.000000	7.049608	1.101828	1995.000000	5.208877	34.470000	-119.670000
9313	4.750000	1.191810	3.558190	3.729200	6.000000	4.583333	1.083333	69.000000	2.875000	37.800000	-121.290000
mean_sample	2.066012	2.142076	0.529594	3.871337	28.619864	5.445038	1.099305	1422.299612	3.113793	35.632167	-119.565215
median_sample	1.785000	1.919566	0.423880	3.522700	29.000000	5.236540	1.047404	1165.000000	2.823360	34.260000	-118.510000
max_sample	5.000010	4.816358	3.806190	15.000100	52.000000	141.909091	25.636364	28566.000000	1243.333333	41.950000	-114.490000
min_sample	0.149990	-0.566310	0.000060	0.499900	1.000000	0.846154	0.375000	5.000000	0.692308	32.540000	-124.300000

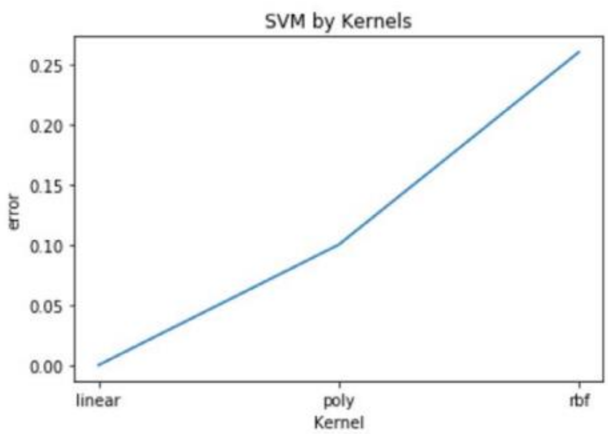
The reasons of these failure cases:The above table show the summary of failure cases. The model gives the high value of absolute error when the house value is equal to the maximum value of sample. The model underpredict the value of house. Especially, when the value of MedInc and AveRooms is small (close to the minimum value), the absolute value of error tends to be very high. In contrast to other features, the value of failure case is not significantly different from mean and median of samples suggesting that those features are not a main source of large MAE.

PART -II

In this mini-project we use 6 features of a crab to determine if the crab is either male or female. There are 200 samples with gender labels provided. We start by loading in the data and split the samples into two even subsets of length 100. One dataset will be used for training and validation, while the other is used for testing. After that, I randomly divide the first dataset into two even subsets of length 50. One is used from training and the other is used for validation. I then consider different weighting parameters with a linear kernel within an SVM model, in order to determine which weighting parameter yields the lowest error. We apply this model to the validation set. The result is show in the plot below.



We can see that the lowest error occurs between approximately 5.5 and 7.3. Given this, I choose the weighting parameter value of 6 and then test for different kernel types. Again, I apply the models to the validation set. The result is given by the plot below.



We see that the linear kernel provides the lowest error in this case and therefore, I get the best model (linear kernel and weighting value = 6).

After trying different value of C and gamma with each kernel type (linear, polynomial and rbf), I get the best model which is the linear kernel with C value of 6 (best choice since it gives lowest error and no overfitting). I take this best model and apply it over the testing set. I yield the following evaluation metrics:

```
## step 5 evaluate your results in terms of accuracy, real, or precision.

#############placeholder 5: metrics ######################
# func_confusion_matrix is provided in conf_matrix.py
# You might re-use this function from previous homework assignment.
y_pred = model.predict(X_test)
conf_matrix, accuracy, recall_array, precision_array = func_confusion_matrix(Y_test, y_pred)

print("Confusion Matrix: ")
print(conf_matrix)
print("Average Accuracy: {}".format(accuracy))
print("Per-Class Precision: {}".format(precision_array))
print("Per-Class Recall: {}".format(recall_array))

#############placeholder end #######################
```

```
Confusion Matrix:
[[44  4]
 [ 1 51]]
Average Accuracy: 0.95
Per-Class Precision: [0.97777778 0.92727273]
Per-Class Recall: [0.91666667 0.98076923]
```

We see that the accuracy is quite high and therefore our model performed well.

The precision rate of y =1 is 0.9777778. The precision rate of y =-1 is 0.92727272. The recall rate of y = 1 is 0.91666667. The recall rate of y =-1 is 0.98076923.

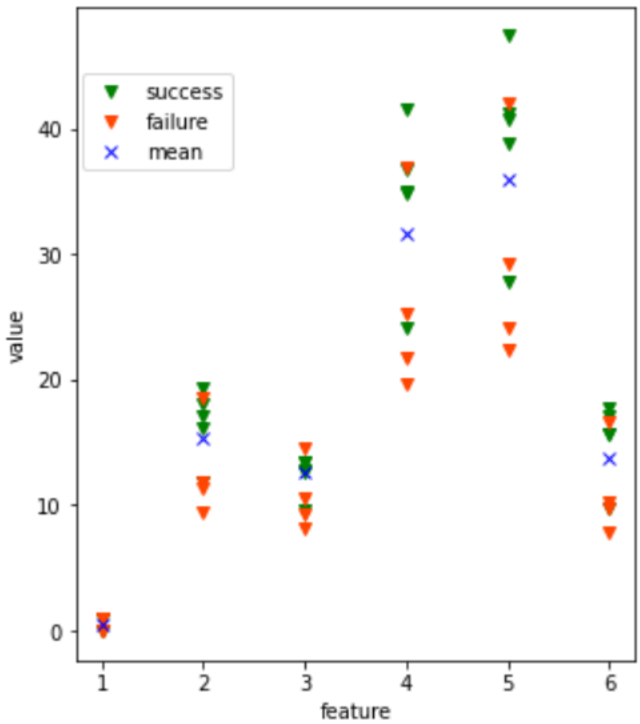
Finally, I created two for loops that would print out both the successful cases and failure cases (5) from the data. I have provided a screenshot of the successful examples and all the failure examples below.

```
Successful examples
Y_actual: -1.0 Y_hat: -1.0 Feature: [ 0.   18.   13.4 36.7 41.3 17.1]
Y_actual: -1.0 Y_hat: -1.0 Feature: [ 1.   19.3 13.5 41.6 47.4 17.8]
Y_actual: -1.0 Y_hat: -1.0 Feature: [ 1.   11.8  9.6 24.2 27.8  9.7]
Y_actual: -1.0 Y_hat: -1.0 Feature: [ 0.   17.1 12.6 35.  38.9 15.7]
Y_actual: -1.0 Y_hat: -1.0 Feature: [ 1.   16.1 12.8 34.9 40.7 15.7]


Failure examples
Y_actual: 1.0 y_hat: -1.0 Feature: [ 1.   9.5  8.2 19.6 22.4  7.8]
Y_actual: 1.0 y_hat: -1.0 Feature: [ 0.  18.5 14.6 37.  42.  16.6]
Y_actual: -1.0 y_hat: 1.0 Feature: [ 1.  11.8 10.5 25.2 29.3 10.3]
Y_actual: 1.0 y_hat: -1.0 Feature: [ 0.  11.4  9.2 21.7 24.1  9.7]
Y_actual: 1.0 y_hat: -1.0 Feature: [ 0.  15.  12.3 30.1 33.3 14. ]
```

In this case, I have also visualized my success and failure examples (both by tabulating and graphically):

	Y_actual	Y_hat	f1	f2	f3	f4	f5	f6
Sucess	-1	-1	0.00	18.000	13.400	36.700	41.300	17.10
Sucess	-1	-1	1.00	19.300	13.500	41.600	47.400	17.80
Sucess	-1	-1	1.00	11.800	9.600	24.200	27.800	9.70
Sucess	-1	-1	0.00	17.100	12.600	35.000	38.900	15.70
Sucess	-1	-1	1.00	16.100	12.800	34.900	40.700	15.70
Failure	1	-1	1.00	9.500	8.200	19.600	22.400	7.80
Failure	1	-1	0.00	18.500	14.600	37.000	42.000	16.60
Failure	-1	1	1.00	11.800	10.500	25.200	29.300	10.30
Failure	1	-1	0.00	11.400	9.200	21.700	24.100	9.70
Failure	1	-1	0.00	15.000	12.300	30.100	33.300	14.00
mean	-	-	0.55	15.287	12.578	31.578	35.901	13.72
median	-	-	1.00	15.350	12.600	31.750	36.500	13.80



Analyzing the returns:According to the above figure, the model cannot predict well the examples with very small value of front-allip, rearwidth, length, width and depth. Also, as observed, if the species is equal to 1, the tendency of model to incorrectly classify the example becomes higher.