Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

536 : APPLIED MACHINE LEARNING

PROJECT 2 : Logistic Regression, SVM and Neural Networks

Project Report

Abstract:

The objective of this project is to understand the workflow of supervised learning algorithms by utilizing logistic regression, support vector machine and neural nets. Evaluate the model's performance using confusion matrix, accuracy, classification report etc. In this project, California housing dataset is used in which median house value is predicted/classified as expensive or not expensive (Binary classification problem) based on the 8 numeric attributes (independent variables). The above mentioned machine learning algorithms are applied on the dataset with various combinations of the hyperparameters of each model. Out of all the models and their evaluations, it is concluded that Neural Network with parameters 'adam' optimizer, 'binary_crossentropy' loss function with 2 hidden layers, 30 neurons each has worked well with the dataset selected with high accuracy of 86.43%.

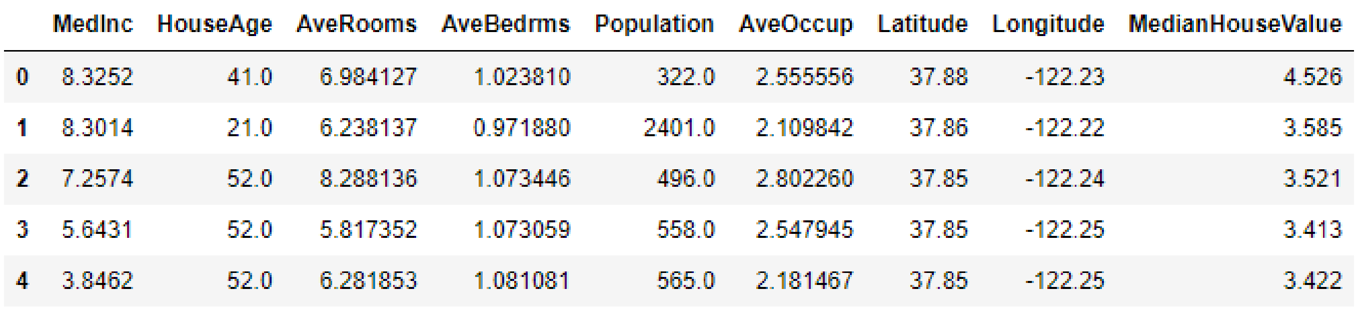
Programming Language: Python

1. Introduction:

Characteristics of the dataset:

- The California housing dataset is imported from scikit-learn library. It was originally derived from the 1990 U.S. census, using one row per census block group.
- The dataset consists of total 20640 instances and 9 attributes - median income in block, median house age in block, average number of rooms, average number of bedrooms, block population, average house occupancy, house block latitude, house block longitude and median house value. Among these, MedianHouseValue is a target variable (dependent variable) and remaining are independent variables.
- There are no missing attribute values and no null values in the dataset.
- No duplicate rows are present in the dataset and hence no data leakage problem.
- Class Distribution is 8709 rows in Expensive class and 11931 rows in Not Expensive class. We can say that the dataset is not biased towards any class and the dataset has 20640 instances, 9 attributes with no null values and no duplicate rows which makes the dataset is appropriate and worth to explore.

Below figure shows the sample data.



	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedianHouseValue
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

Outline approaches taken to solve the problem:

- Considering the logistic regression, SVM and Multi-Layer Perception (Neural Network) models' applications, a new column(target feature) has been created in the dataset to apply supervised machine learning model to classify the median house value as Expensive or Not Expensive.
- MedianHouseValue above $200000 are categorized as Expensive (1-True for Neural Network) and less than or equal to $200000 are categorized as Not Expensive (0-False for Neural Network).

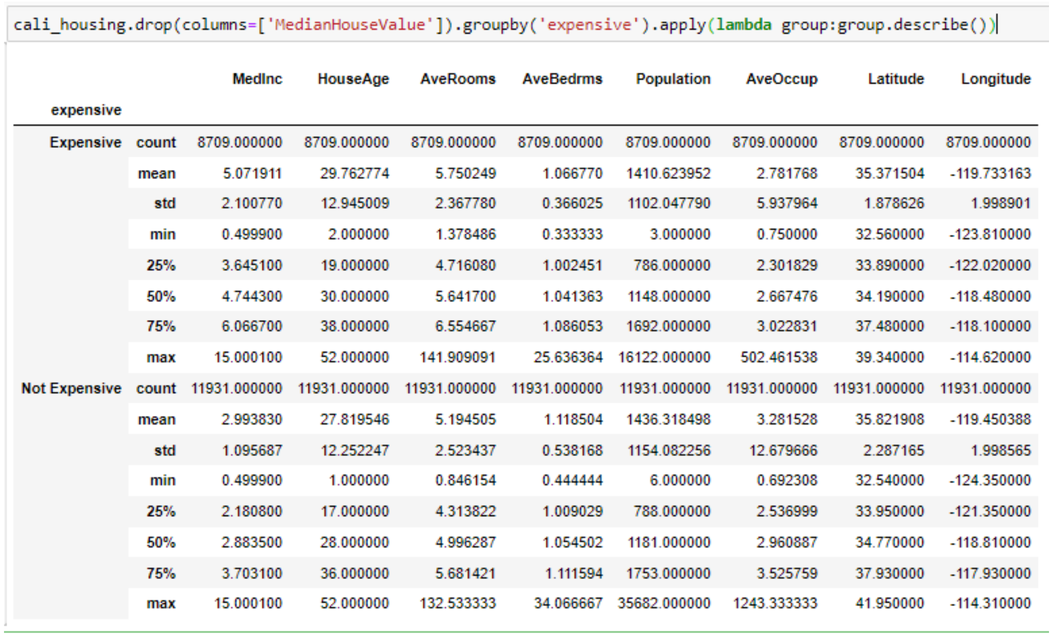
  cali_housing['expensive'] = np.where(cali_housing['MedianHouseValue']>2,"Expensive","Not Expensive")

- Before applying any machine learning model, feature scaling has been done and the dataset is divided into training and testing datasets in the ratio 75:25 respectively.

2. Statistical Summary of the data:

Statistical summary of attributes with respect to each class:

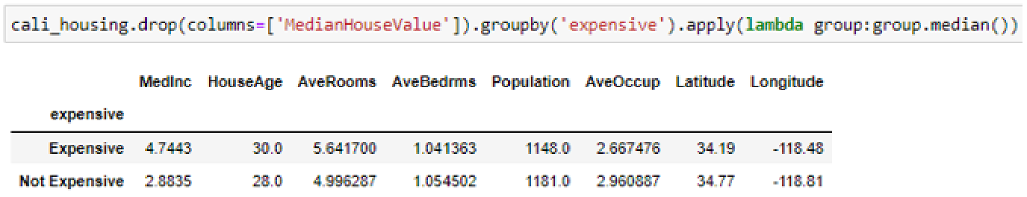
Target (dependent) variable – Expensive / Not Expensive



```
cali_housing.drop(columns=['MedianHouseValue']).groupby('expensive').apply(lambda group:group.describe())
```

expensive		MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
Expensive	count	8709.000000	8709.000000	8709.000000	8709.000000	8709.000000	8709.000000	8709.000000	8709.000000
	mean	5.071911	29.762774	5.750249	1.066770	1410.623952	2.781768	35.371504	-119.733163
	std	2.100770	12.945009	2.367780	0.366025	1102.047790	5.937964	1.878626	1.998901
	min	0.499900	2.000000	1.378486	0.333333	3.000000	0.750000	32.560000	-123.810000
	25%	3.645100	19.000000	4.716080	1.002451	786.000000	2.301829	33.890000	-122.020000
	50%	4.744300	30.000000	5.641700	1.041363	1148.000000	2.667476	34.190000	-118.480000
	75%	6.066700	38.000000	6.554667	1.086053	1692.000000	3.022831	37.480000	-118.100000
	max	15.000100	52.000000	141.909091	25.636364	16122.000000	502.461538	39.340000	-114.620000
Not Expensive	count	11931.000000	11931.000000	11931.000000	11931.000000	11931.000000	11931.000000	11931.000000	11931.000000
	mean	2.993830	27.819546	5.194505	1.118504	1436.318498	3.281528	35.821908	-119.450388
	std	1.095687	12.252247	2.523437	0.538168	1154.082256	12.679666	2.287165	1.998565
	min	0.499900	1.000000	0.846154	0.444444	6.000000	0.692308	32.540000	-124.350000
	25%	2.180800	17.000000	4.313822	1.009029	788.000000	2.536999	33.950000	-121.350000
	50%	2.883500	28.000000	4.996287	1.054502	1181.000000	2.960887	34.770000	-118.810000
	75%	3.703100	36.000000	5.681421	1.111594	1753.000000	3.525759	37.930000	-117.930000
	max	15.000100	52.000000	132.533333	34.066667	35682.000000	1243.333333	41.950000	-114.310000

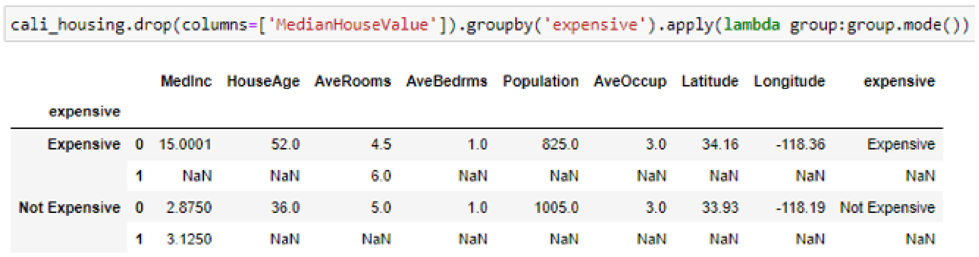
Calculation of Median with respect to each class:



```
cali_housing.drop(columns=['MedianHouseValue']).groupby('expensive').apply(lambda group:group.median())
```

expensive	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
Expensive	4.7443	30.0	5.641700	1.041363	1148.0	2.667476	34.19	-118.48
Not Expensive	2.8835	28.0	4.996287	1.054502	1181.0	2.960887	34.77	-118.81

Calculation of Mode with respect to each class:

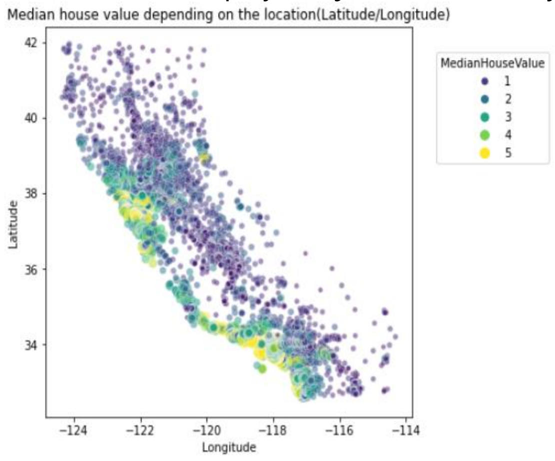


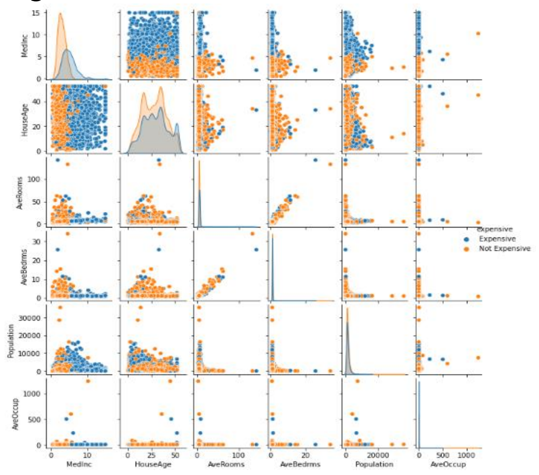
```
cali_housing.drop(columns=['MedianHouseValue']).groupby('expensive').apply(lambda group:group.mode())
```

expensive		MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	expensive
Expensive	0	15.0001	52.0	4.5	1.0	825.0	3.0	34.16	-118.36	Expensive
	1	NaN	NaN	6.0	NaN	NaN	NaN	NaN	NaN	NaN
Not Expensive	0	2.8750	36.0	5.0	1.0	1005.0	3.0	33.93	-118.19	Not Expensive
	1	3.1250	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Insights from statistics:

- From mode calculation we can see that two rows are displayed because of 2 mode values i.e. AveRooms of 6.0 and 4.5 have equal frequency in Expensive class. Similarly MedInc of 2.8750 and 3.1250 in Not Expensive class.
- From the summary statistics, it is observed that there is a huge difference between max and 75 percentile values of features MedInc, HouseAge, AveRooms, AveBedrms, Population and AveOccup which indicates that there are few extreme values in the dataset.
  Note: 25 and 75 percentile values indicate border of the upper and lower quarter of the data respectively.
- It is observed that the mean is higher than the median for the features MedInc, AveRooms, AveBedrms, Population and AveOccup which indicates that data is right skewed. Skewness is a measure of asymmetry of the probability distribution of a real-valued random variable about its mean.
- From the below scatter plot, it can be observed that expensive houses (yellow color) are located along the coast (considering geographical map of California).
- Median Income has the highest correlation value 0.68 which indicates that it is strongly linked to Median House Value. From pairplot visualization also, we can clearly see Median Income (first row and first column) plays major role in classifying Median House Value.





3. Methods:

Logistic Regression:

Logistic regression is a supervised classification algorithm which measures the relationship between one or more independent variables and the categorical dependent variable by estimating probabilities using a logistic (sigmoid) function.

Parameters used:

- Logistic regression is a linear classifier, so we will use a linear function $f(\mathbf{x}) = b_0 + b_1x_1 + \cdots + b_rx_r$, also called the logit. The variables b_0, b_1, ..., b_r are the estimators of the regression coefficients, which are also called the predicted weights or just coefficients.
- Logistic regression determines the best predicted weights b_0, b_1, ..., b_r. To get the best weights, log-likelihood function (LLF) is used and maximized for all observations i = 1, ..., n. This method is called the maximum likelihood estimation (MLE).
- In this project 'liblinear' solver parameter is passed to the model.

lr = LogisticRegression(solver='liblinear')

Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

- The best accuracy we got for logistic regression model is 82.71%. Below are the Coefficients and intercept values as well as training and testing set scores.

  lr.coef_: [[-2.49964423 -0.33460648 0.85292619 -1.00790406 -0.06542772 2.67512768 3.63493894 3.43565408]]
  lr.intercept_: [0.62545825]
  Training set score: 83.31
  Test set score: 82.71

- From the above result, it can be observed that b_0 is present in one dimensional array while other coefficients b_1, $b_{2,}$ …. are present in 2-dimensional array.

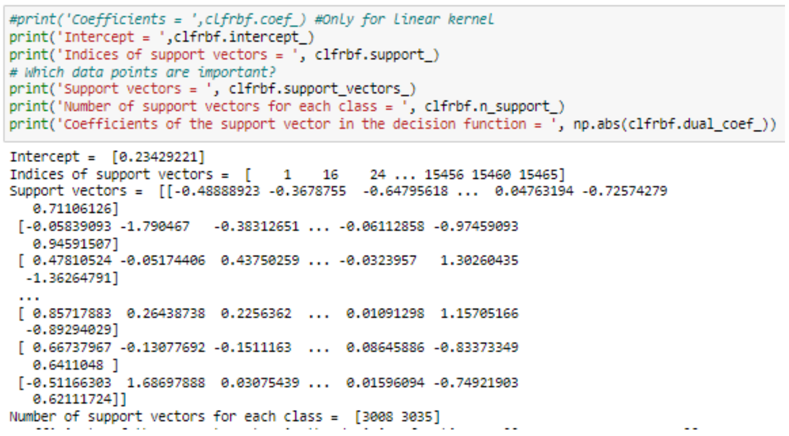
Support Vector Machine (SVM):

In Support Vector Machine (SVM) algorithm, the objective is to find a hyperplane in an N-dimensional space (here N=number of features) that distinctly classifies the data points. Out of the many possible hyperplanes to separate the two classes of data points, our objective is to find a plane that has the maximum margin (distance between data points of both classes). Maximizing the margin distance provides room for incoming future data points, which can be classified with more confidence. This can be done using support vectors.

Parameters used:

- In SVM, the hyperparameters C and kernel have major role.
- For this dataset, with change in C value, there is not much difference in accuracy. For example when C = 100, we got 84.36% accuracy and with C = 10, we got 84.32% and for remaining C values also we observed that the result is almost same.
- Among all the kernels linear, rbf, poly, sigmoid, precomputed best accuracy we got is 85.54% for rbf-gaussian kernel.
- Like logistic regression, in SVM also coefficients and intercepts are calculated and the results are as shown below.

  SVM rbf kernel Accuracy: 85.54263565891473
  Training set score: 86.28
  Test set score: 85.54



```
#print('Coefficients = ',clfrbf.coef_) #Only for linear kernel
print('Intercept = ',clfrbf.intercept_)
print('Indices of support vectors = ', clfrbf.support_)
# Which data points are important?
print('Support vectors = ', clfrbf.support_vectors_)
print('Number of support vectors for each class = ', clfrbf.n_support_)
print('Coefficients of the support vector in the decision function = ', np.abs(clfrbf.dual_coef_))
```

```
Intercept =  [0.23429221]
Indices of support vectors =  [    1    16    24 ... 15456 15460 15465]
Support vectors =  [[-0.48888923 -0.3678755  -0.64795618 ...  0.04763194 -0.72574279
   0.71106126]
 [-0.05839093 -1.790467   -0.38312651 ... -0.06112858 -0.97459093
   0.94591507]
 [ 0.47810524 -0.05174406  0.43750259 ... -0.0323957   1.30260435
  -1.36264791]
 ...
 [ 0.85717883  0.26438738  0.2256362  ...  0.01091298  1.15705166
  -0.89294029]
 [ 0.66737967 -0.13077692 -0.1511163  ...  0.08645886 -0.83373349
   0.6411048 ]
 [-0.51166303  1.68697888  0.03075439 ...  0.01596094 -0.74921903
   0.62111724]]
Number of support vectors for each class =  [3008 3035]
```

Neural Network:

Neural Networks are multi-layer networks of neurons that are used to classify the data and make predictions. Each neuron takes inputs, does some math using activation function and produces one output. The activation function is used to turn an unbounded input into an output that has predictable form. A commonly used activation function is sigmoid function.

Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

Parameters used:

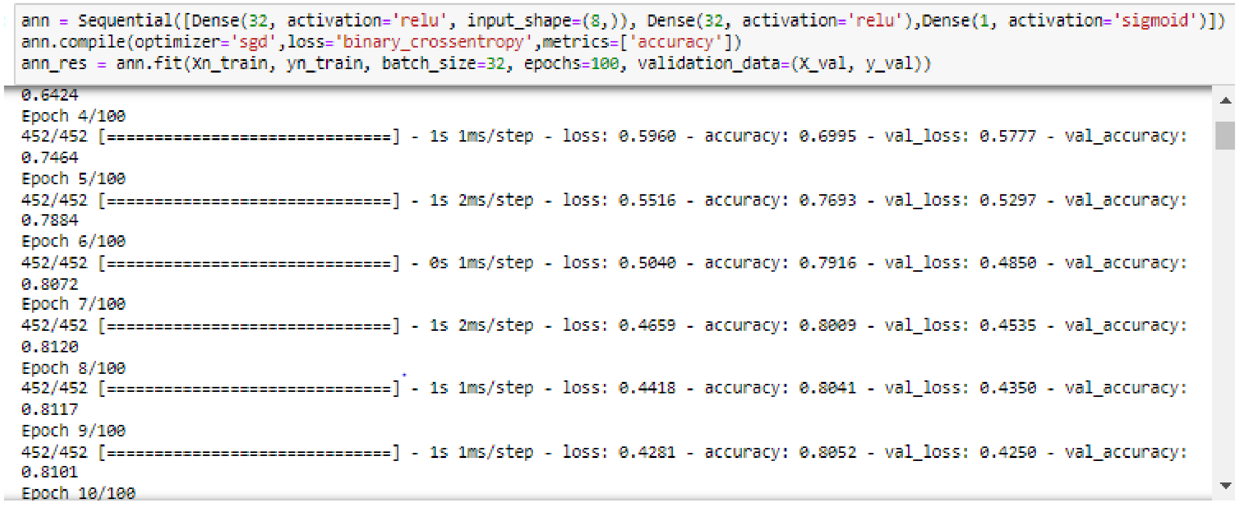
- Now dataset is split into training and training datasets. Here validation datasets are also created. Validation dataset is used to evaluate the models to tune hyperparameters and test dataset is used to estimate the accuracy of the final model.
- Neural Network Architecture:

  8 input features (size 8 neurons) → Hidden Layer 1 (size 32 neurons) → Hidden Layer 2 (size 32 neurons) → Output (size 1 neuron)
- Activation function : ReLU at hidden layers and Sigmoid function at output.
- Keras sequential model is used to describe the neural network layers. In the below code snippet, Dense indicates that it is a fully-connected layer.

  ann = Sequential([Dense(32, activation='relu', input_shape=(8,)), Dense(32, activation='relu'), Dense(1, activation='sigmoid'),])
- Now the model is compiled by giving parameters such as optimizer, loss function and accuracy metric. In the below code snippet, 'sgd' means stochastic gradient descent and 'binary_crossentropy' loss function for output takes the values 1 or 0.

  ann.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
- Model is applied on the training dataset. Epochs is declared which indicates the size of our mini-batch and how long we want to train the dataset. Validation dataset is passed so that the model displays the results of the validation dataset at each point.

  ann_res = ann.fit(Xn_train, yn_train, batch_size=32, epochs=100, validation_data=(X_val, y_val))
- Model training at each step: Here we can observe that accuracy is getting increased and loss is getting decreased with each Epoch.



```
ann = Sequential([Dense(32, activation='relu', input_shape=(8,)), Dense(32, activation='relu'),Dense(1, activation='sigmoid')])
ann.compile(optimizer='sgd',loss='binary_crossentropy',metrics=['accuracy'])
ann_res = ann.fit(Xn_train, yn_train, batch_size=32, epochs=100, validation_data=(X_val, y_val))
```

```
0.6424
Epoch 4/100
452/452 [==============================] - 1s 1ms/step - loss: 0.5960 - accuracy: 0.6995 - val_loss: 0.5777 - val_accuracy:
0.7464
Epoch 5/100
452/452 [==============================] - 1s 2ms/step - loss: 0.5516 - accuracy: 0.7693 - val_loss: 0.5297 - val_accuracy:
0.7884
Epoch 6/100
452/452 [==============================] - 0s 1ms/step - loss: 0.5040 - accuracy: 0.7916 - val_loss: 0.4850 - val_accuracy:
0.8072
Epoch 7/100
452/452 [==============================] - 1s 2ms/step - loss: 0.4659 - accuracy: 0.8009 - val_loss: 0.4535 - val_accuracy:
0.8120
Epoch 8/100
452/452 [==============================] - 1s 1ms/step - loss: 0.4418 - accuracy: 0.8041 - val_loss: 0.4350 - val_accuracy:
0.8117
Epoch 9/100
452/452 [==============================] - 1s 1ms/step - loss: 0.4281 - accuracy: 0.8052 - val_loss: 0.4250 - val_accuracy:
0.8101
Epoch 10/100
```

- Different combinations of neurons are implemented and finalized 30 neurons in each hidden layer with epoch size 100, batch size 5 and we got an accuracy of 86.43%.
- Two different gradient descent solvers are used in this project. They are Stochastic Gradient Descent (sgd) and Adam Gradient Descent. We got best accuracy with adam gradient descent after trying out different combinations using grid search.

Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

4. Results:

Classification report:

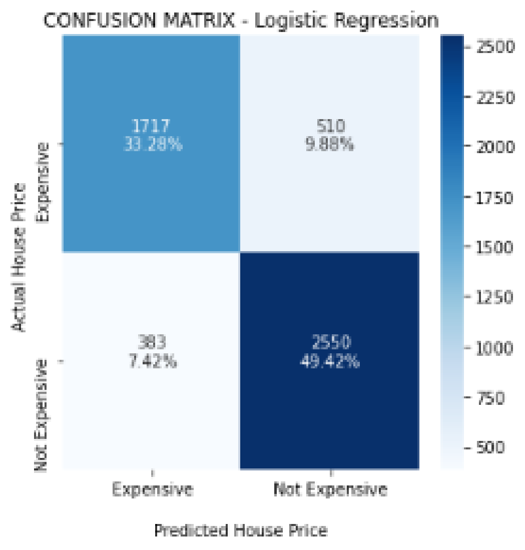
Logistic Regression

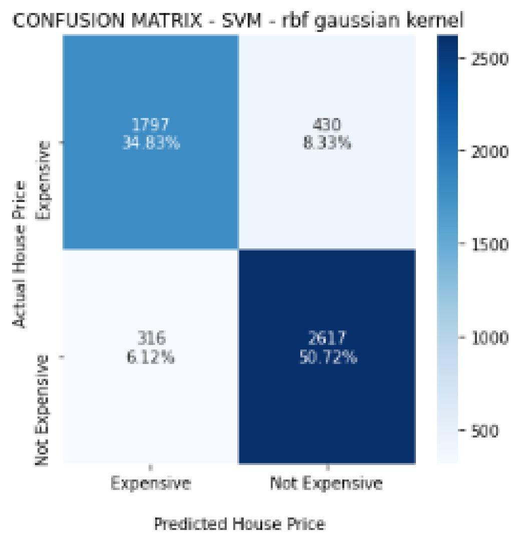
```
               precision    recall  f1-score   support

    Expensive       0.82      0.77      0.79      2227
Not Expensive       0.83      0.87      0.85      2933

     accuracy                           0.83      5160
    macro avg       0.83      0.82      0.82      5160
 weighted avg       0.83      0.83      0.83      5160
```

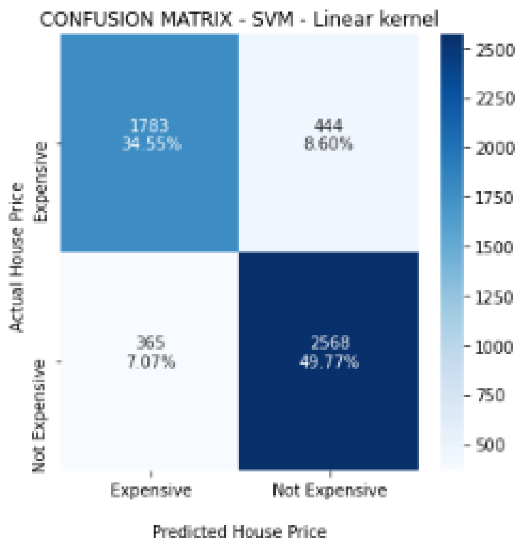
Support Vector Machine:

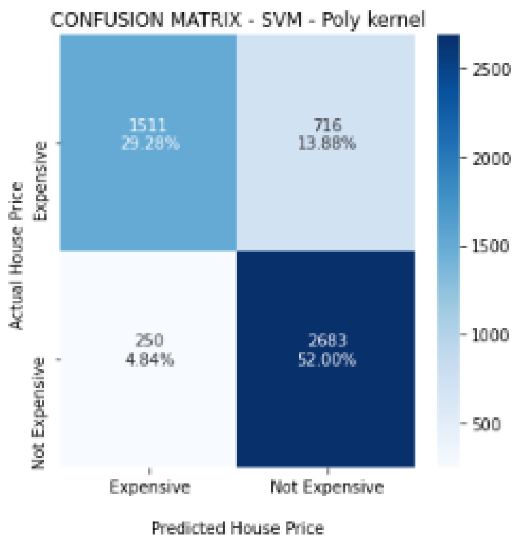
```
               precision    recall  f1-score   support

    Expensive       0.85      0.81      0.83      2227
Not Expensive       0.86      0.89      0.88      2933

     accuracy                           0.86      5160
    macro avg       0.85      0.85      0.85      5160
 weighted avg       0.86      0.86      0.85      5160
```

Confusion Matrix:

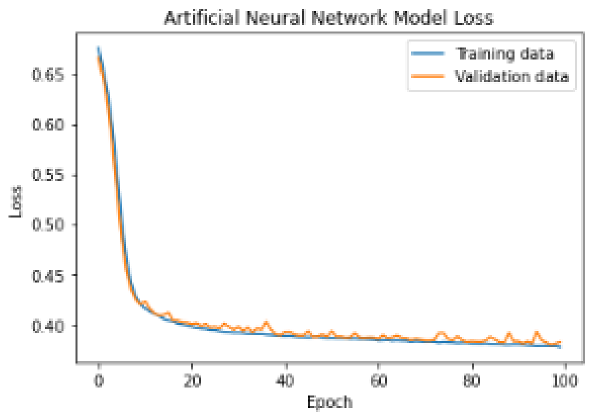


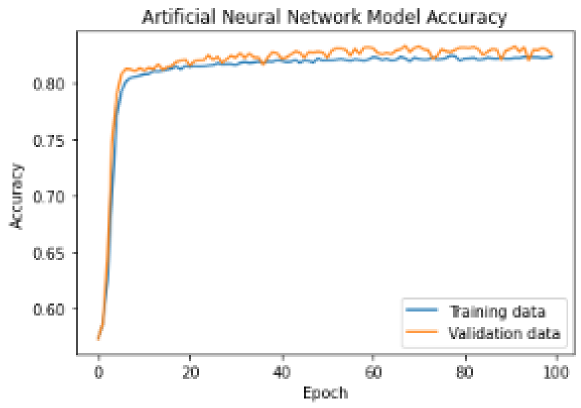






Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263





5. Discussions:

The accuracies of all the models are summarized as below.

Logistic Regression Accuracy – 82.71%

K-fold cross validation on logistic regression accuracy – 79.35%

SVM model Accuracy –

SVM algorithm kernel	Other param, C – Regularization parameter	Accuracy	Training set score	Testing set score
linear	C = 1	84.32%	84.33%	84.32%
poly	degree=1, gamma=auto, C=0.7	81.28%	82.13%	81.28%
rbf	gamma=0.6, C=0.8	85.54%	86.28%	85.54%
sigmoid		71.16%	70.72%	71.16%
precomputed		Precomputed kernel cannot be implemented because the input must be a square matrix and our input dataset is a 15480 X 8 matrix.		

Artificial Neural Network model Accuracy –

Stochastic Gradient Descent optimizer – 81.91%

Adam Gradient Descent – 86.43%

- From the above data it can be observed that ANN model can classify any new input feature with 86.43% accuracy.
- SVM model with Gaussian rbf kernel also has good accuracy with 85.54%.
- In my opinion, ANN model is expected to work best for this dataset because of the deep architecture present in the model and results also show the same thing.
- When we compare all the models, ANN model works best for this dataset.

6. Conclusion:

- Each model worked well in predicting the Median House Value as expensive or not expensive.
- When a coastal house is given with corresponding features including latitude and longitude, the models classified correctly that the house is expensive.
- Neural Network has the best accuracy currently for this dataset.
- The accuracy can be improved with different combination of parameters in each model.
- Further rigorous testing is needed before moving any machine learning model into production.

Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

7. (Graduate student) **Detailed description of each model:**

Logistic Regression:

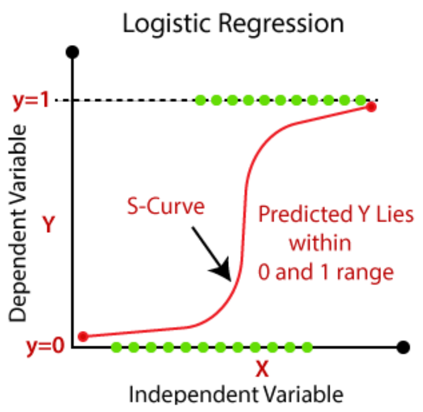
Logistic regression is a supervised classification algorithm which measures the relationship between one or more independent variables and the categorical dependent variable by estimating probabilities using a logistic (sigmoid) function. Here the independent variables can be numeric or categorical variables, but the dependent variable will always be categorical.

In simple terms Logistic regression is a predictive analysis algorithm and based on the concept of probability. Sigmoid function is used to map predictions to probabilities.

Example: For binary regression, the conditional probability of the dependent variable Y, given independent variable X is calculated as below.

P(Y=1|X) or P(Y=0|X)

The above can be read as the conditional probability of Y = 1, given X or conditional probability of Y = 0, given X. P(Y|X) is approximated as a sigmoid function applied to a linear combination of input features.



- In logistic regression, output of linear function is taken and flatten its value within the range of [0,1] using the sigmoid function. If the compressed value is greater than 0.5 , we assign label 1 to it, else label 0 will be assigned.
- The regression coefficients/parameters ($b_{0,}$ $b_{1,......,}b_p$) are estimated using Maximum Likelihood Estimation(MLE), which finds the set of parameters for which the probability of the observed data is greatest.

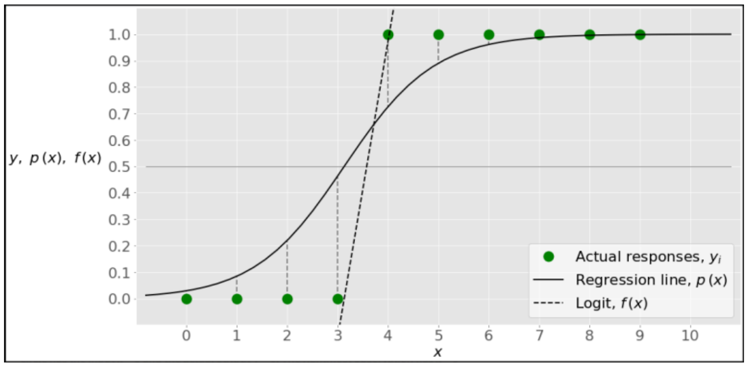
Application of Logistic regression on a dataset:

Logistic regression is applied on the dataset when below conditions are observed.

- Outliers are not present in the data. An outlier is a data point in a dataset that lies outside the overall distribution of the dataset. It can be identified by analyzing the independent variables.
- No correlation or multi-collinearity between the independent variables.

Single-Variate logistic Regression:

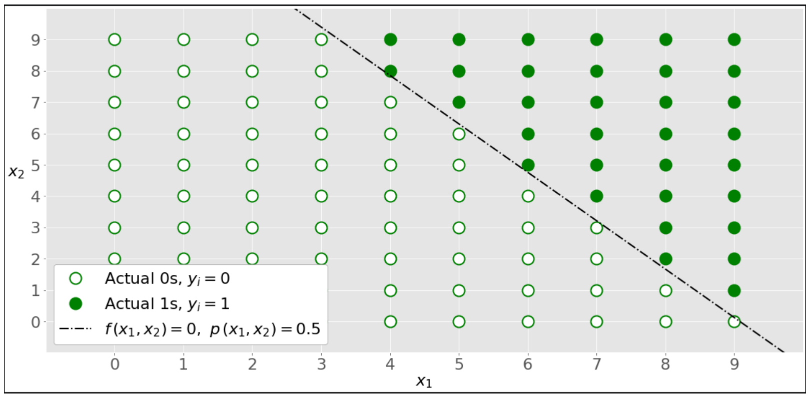
Single-variate logistic regression is the most straightforward case of logistic regression. There is only one independent variable (or feature), which is $\mathbf{x} = x$.



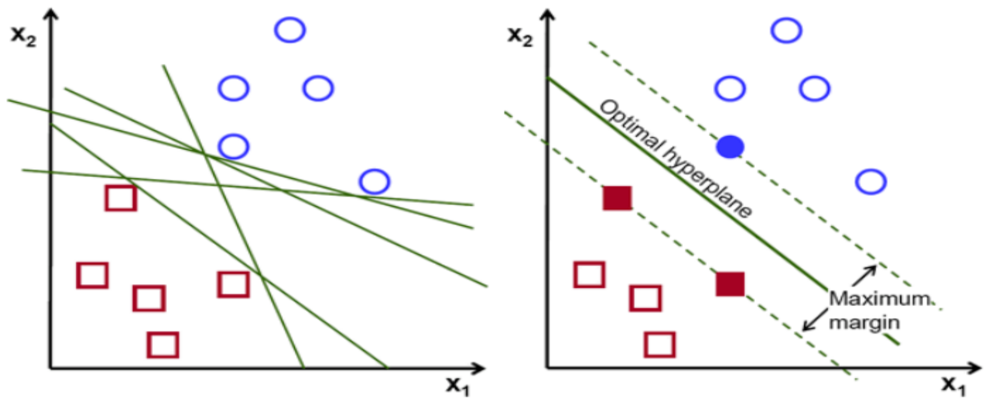
Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

Multi-Variate Logistic Regression:

Multi-variate logistic regression has more than one input variable. This figure shows the classification with two independent variables, x_1 and x_2.



Support Vector Machine:
In Support Vector Machine (SVM) algorithm, the objective is to find a hyperplane in an N-dimensional space (here N=number of features) that distinctly classifies the data points.

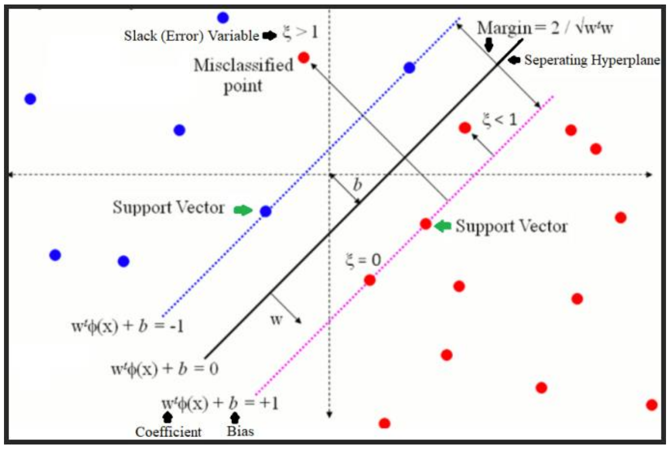


Possible hyperplanes

Out of the many possible hyperplanes to separate the two classes of data points, our objective is to find a plane that has the maximum margin (distance between data points of both classes). Maximizing the margin distance provides room for incoming future data points, which can be classified with more confidence. This can be done using support vectors.

- Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. These points are present within the soft margin (distance between the observations and the threshold).
- The dimension of hyperplane depends on the number of features. For example, 2 features dataset has line hyperplane and 3 features dataset has 2-dimensional plane as hyperplane.
- In SVM algorithm, linear function output is taken and if that output is greater than 1, we identify it with one class and if the output is -1, we identify it with another class. Since the threshold values are changed to 1 and -1 in SVM, this reinforcement range of values [-1,1] acts as a margin.
- There are different types of SVM kernels. Out of them, polynomial and RBF are especially useful when the data points are not linearly separable.
- Maximal Margin classifiers are super sensitive to outliers in the training data. We can make it better by allowing misclassifications. When we allow misclassification, the distance between the observations and the threshold is called Soft Margin.

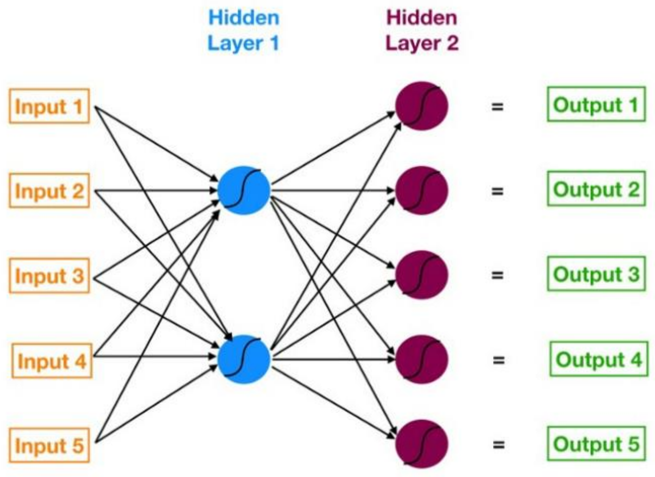
Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263



- Cross validation techniques are used to determine the numbers of misclassifications and observations to allow inside of the soft margin to get the best classification. The name support vector classifier comes from the fact the observations on the edge and within the soft margin are called support vectors.
- The kernel SVM projects the non-linearly separable data lower dimensions to linearly separable data in higher dimensions in such a way that data points belonging to different classes are allocated to different dimensions.
- Linear SVM: Linear SVM is less prone to overfitting than non-linear. If the number of features is large compared to training sample, we use linear kernel. If the number of features is small but the training sample is large may also need linear kernel but we should try to add more features; if feature number is small (10^0 - 10^3), and the sample number is intermediate (10^1 - 10^4), use Gaussian kernel will be better.
- Gaussian Kernel: It is a general-purpose kernel used when there is no prior knowledge about the data.

Artificial Neural Networks:

Neural Networks are multi-layer networks of neurons that are used to classify the data and make predictions. Each neuron takes inputs, does some math using activation function and produces one output. The activation function is used to turn an unbounded input into an output that has predictable form. A commonly used activation function is sigmoid function.



Neural network with two hidden layers

Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

- The arrows that connect the dots shows how all the neurons are interconnected and how data travels from the input layer all the way through the output layer. The neural network learns from its mistake using a process known as backpropagation.
- In neural network, changing the weight of any connection has a reverberating effect across all the other neurons and their activations in the subsequent layers. This is because each neuron in a neural network is like its own model.

$$\begin{bmatrix} W1,1 & W2,1 & W3,1 & W4,1 & W5,1 \\ W1,2 & W2,2 & W3,2 & W4,2 & W5,2 \end{bmatrix} \times \begin{bmatrix} X1 \\ X2 \\ X3 \\ X4 \\ X5 \end{bmatrix} + \begin{bmatrix} Bias1 \\ Bias2 \end{bmatrix} = \begin{bmatrix} Z1 \\ Z2 \end{bmatrix}$$

- Artificial Neural Network is a computing system made up of a number of simple, highly interconnected processing elements which process information by their dynamic state response to external inputs.
- The magnitude of the error of a specific neuron **(relative to the errors of all the other neurons)** is directly proportional to the impact of that neuron's output (a.k.a. activation) on our cost function**.**
- With neural networks, the process is very similar: you start with some random weights and bias vectors, make a prediction, compare it to the desired output, and adjust the vectors to predict more accurately the next time. The process continues until the difference between the prediction and the correct targets is minimal.
- Knowing when to stop the training and what accuracy target to set is an important aspect of training neural networks, mainly because of overfitting and underfitting scenarios.

8. References:

✓ https://medium.com/axum-labs/logistic-regression-vs-support-vector-machines-svm-c335610a3d16
✓ https://towardsdatascience.com/understanding-neural-networks-19020b758230
✓ https://towardsdatascience.com/support-vector-machines-explained-with-python-examples-cb65e8172c85