Housing value prediction using Regression models:

Problem description

The dataset used in the experiment comprises of houses in Boston with 13 columns defining characteristics of each house. The last column is for the median value of houses in $1000's. The test is to compare the performances of three regression models- SVM regression, KNN regression, Linear regression. We are using mean_sqaure_error metric to predict the error of the model using 80% data set for training and 20% of data set for testing.

SVM regression Model

SVM is a technique used to separate a set of data into identical classes using a line or a hyperplane (for multi-dimensional data). The following are the parameters of the SVR model:

i. Kernel- It is used to control the dimensionality of the data to reduce the computational cost.

ii. Hyperplane- The line used to compute to predict the continuous output.

iii. Decision Boundary- Is the line of boundary between the positive and negative side of the Hyperplane.

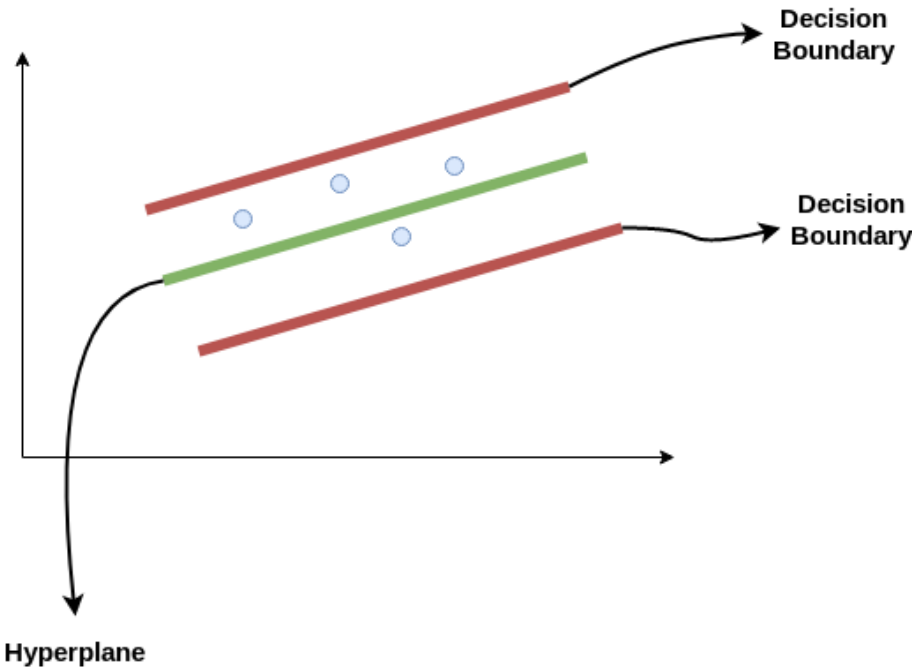

Figure 1 SVR Support Vector Regression Model (Ref.3)

The diagram above shows the two decision boundaries. The purpose of decision boundaries is to encapsulate the maximum number of points so that the hyperplane represents the best fit line. The decision lines are on the positive side (+a distance) and the negative side (-a distance) of the hyperplane. This is good for predicting data of low dimensionality. If the line cannot be used to split the set of data, for instance, the scattered data set then we use a kernel to increase the dimension of data set to split data in 2D or 3D.

KNN (K – Nearest Neighbour) Regression Model

KNN is one of the simplest supervised ML algorithms mostly used for classifying data based on how its neighbours are classified. KNN stores all the available cases and classifies new cases based on a similarity measure. *K* in KNN is a parameter that refers to the number of nearest neighbours to include in the majority voting process. The new data point is classified by majority votes from its nearest neighbours.

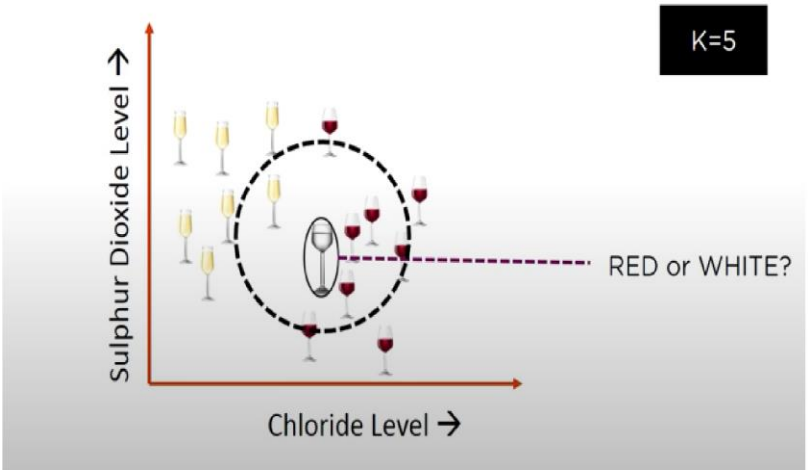

Figure 2 Example of KNN (Ref. 4)

In the above example, the new glass of wine will be classified as red, since 4 out of 5 neighbours are red.

i. When to use KNN:

- Data is labelled.
- Data is noise-free.
- Dataset is small.

ii. Choosing K:

Choosing parameter K is called parameter tuning and is important for better accuracy.

iii. Methods to choose K:

- Sqrt(n), where n is the total number of data points.
- Odd value of K is selected to avoid confusion between two classes of data.

Linear Regression Model

Linear regression is a statistical model used to predict the relationship between independent and dependent variables.

i. When to use Linear Regression:

- The dependent variable *Y* has a linear relationship to the independent variable *X*.

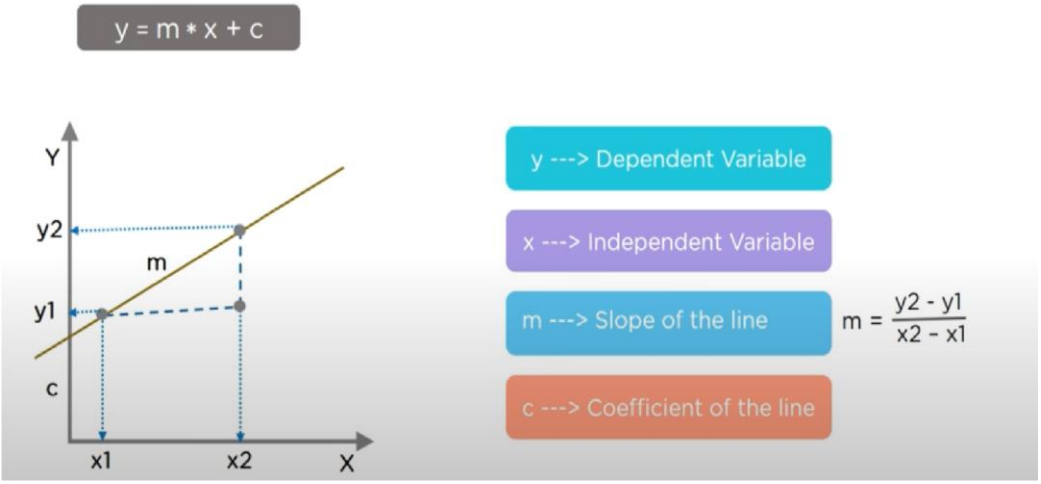

Figure 3 Linear Regression Prediction Model (Ref. 4)

The simplest form of a simple linear regression equation with one dependent and one independent variable is represented by:

y = m * x + c

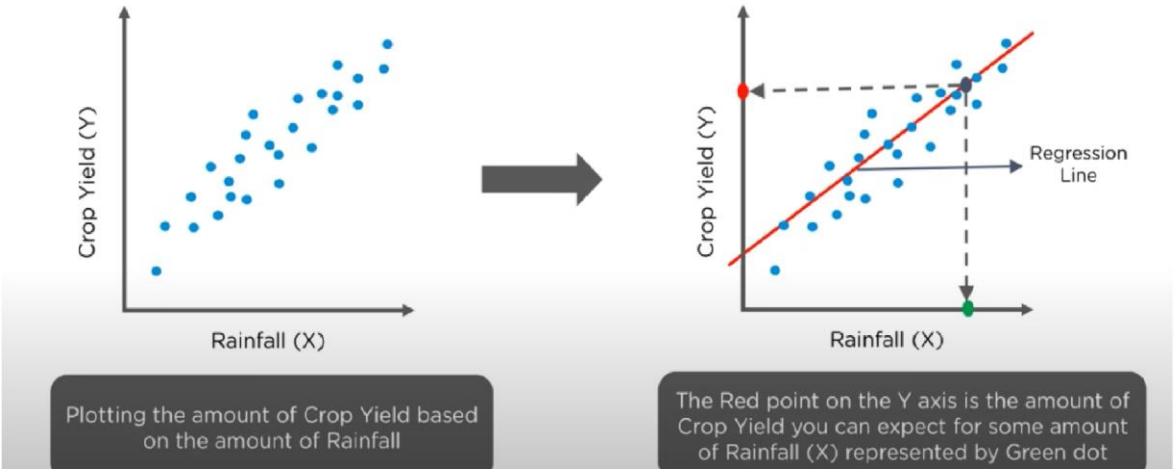

Figure 4 Prediction of crop yield based on the amount of rainfall using the regression line (Ref. 4)

The above experiment shows that how a dependent variable (crop yield) can be predicted using the independent variable (Rainfall).

Experiments with parameter settings

For quantifying the quality of predictions we are using mean_squared_error metrics. The smaller the valu e of mean_squared_error, the better is the performance of the model.

All three models are pre-processed with 'preprocessing.scale()' function to standardize the data set along the axis. Without pre-processing, the models show a significant increase in the mean_squared_error.

With pre-processing function, SVM (keeping other parameters as kernel='rbf', gamma='auto', degree=3, C =1.0, epsilon=0.2, coef0=1) showed 0.2230531091707186 mean_squared_error, KNN (keeping other para meters as n_neighbors=2, metric='minkowski') showed 0.36036888557281466 mean_squared_error, and Linear Regression (keeping other parameters as normalize=True) showed 0.4034193366981747 mean_sq uared_error. However, without pre-processing SVM (81.05175235220372), KNN (47.24031862745099), L inear Regression (34.05648134887462) are the mean_squared_error values which are quite high.

I changed the following parameters to test and find the best performance of the models.

KNN:

1. n_neighbors= 1,2,3…
2. metric= 'euclidean', 'l2', 'minkowski'

Linear regression Model:

1. normalize= True, False

SVM regression Model:

1. kernel: 'linear', 'poly', 'rbf', 'sigmoid'
2. gamma: 'scale', 'auto'
3. coef0: only significant in 'poly' and 'sigmoid' kernels.
4. C= 1,2,3,4….
5. epsilon= 0,1,2…
6. verbose= 0, 1

<u>Observations when changing parameters</u>

Firstly, for KNN I observed that changing the parameter 'metric' did affect the mean_squared_error, how ever changing the 'n_neighbors' to lower and higher values brought a huge difference in the performance . The best performance is at n_neighbors or *K* = 5 which is 0.2546661103054123, the lowest mean_squar ed_error. If I change the *K* = 404 (max value), the mean_squared_error is 0.9572291687728758 is much h igher and is not effective. Similarly keeping it to minimum *K* = 1 also brings the mean_squared_error to 0. 31485363263932997 which is not the best result. We can also use Sqrt(n) and Odd value of *K* methods to determine the best *K* value. If we take the Sqrt(n), where n=506, then we can consider *K* = 22. But this do esn't bring the best result 0.3855504119471646.

In the case of Linear regression Model, the parameter normalize=True/False didn't make any difference t o the mean_squared_error. It stayed at 0.4034193366981747.

Lastly, In SVR method kernel parameter as 'rbf' along with gamma='scale', C=44.0, epsilon=0.2, verbose=0 gives the best performance with mean_squared_error as 0.15125150754176733.

I tried changing kernel to 'linear', 'poly', 'rbf', 'sigmoid'. Sigmoid performed the worst (13.003991385336931), followed by poly (0.5059899677143471)- it improved with coef0=5(0.22957142001548314), linear (0.5041829634575226), and rbf (0.15125150754176733).

I changed the parameter C from 1 to large values (10^10 <) but it stagnates at ≈0.19 mean_square_error. Parameter gamma puts a little effect when changes from 'scale' to 'auto'. The parameter epsilon when 0.2< increases the mean_square_error.

Best performance of Regression Models

According to the experiments conducted by changing the parameters, it is clear that the SVR method performs the best followed by KNN and Linear regression. Although this can change according to the parameter selection. Overall, the SVR seems to show least mean_square_error out of all the models. The results show that SVR can predict with (0.15125150754176733) error in predicting the median value of houses in Boston.

Reference

1. (Simplilearn, 2018)
2. (Sethi)

Bibliography

1. https://scikit-learn.org/. (n.d.). Support Vector Regression (SVR) using linear and non-linear kernels. Retrieved from scikit-learn.org: https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html#sphx-glr-auto-examples-svm-plot-svm-regression-py

2. scikit-learn.org. (n.d.). 3.3. Metrics and scoring: quantifying the quality of predictions. Retrieved from scikit-learn.org: https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error

3. Sethi, A. (n.d.). Support Vector Regression Tutorial for Machine Learning. Retrieved from https://www.analyticsvidhya.com/: https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/

4. Simplilearn. (2018, June 6). KNN Algorithm - How KNN Algorithm Works With Example | Data Science For Beginners | Simplilearn. Retrieved from YouTube: https://www.youtube.com/watch?v=4HKqjENq9OU