Evaluation Metrics With Python Codes

Vishwanath Kulkarni — Published On January 27, 2022 and Last Modified On March 15th, 2022

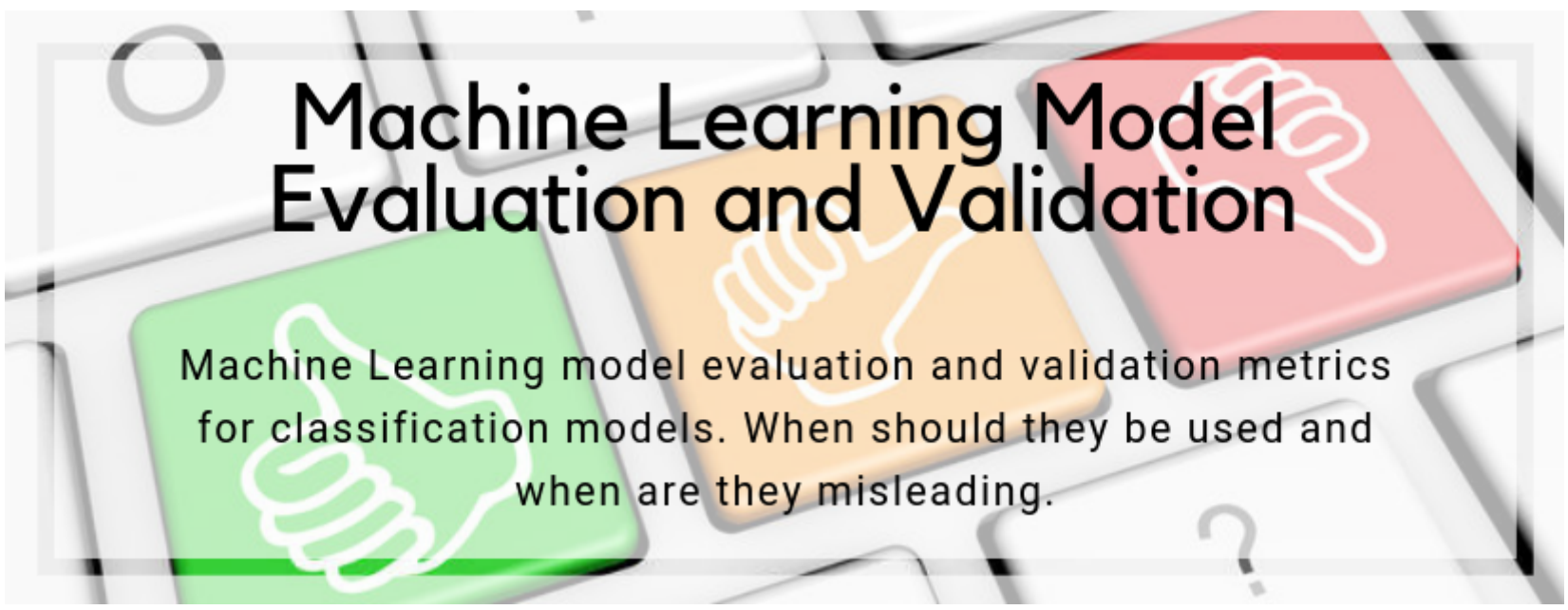
Beginner Machine Learning Python

This article was published as a part of the Data Science Blogathon.

Introduction

The basic idea of building a machine learning model is to assess the relationship between the dependent and independent variables. In doing so, we need to optimize the model performance. There are two types of ML models, classification and regression; for each ML model, we need to optimize for different parameters. Evaluation metrics used for classification problems differ from regression problems. We will go through most of the classification and regression evaluation metrics with the python code to implement them.

Classification Metrics



Source: http://www.easy-analysis.com/category/machine-learning/

Classification models have various evaluation metrics to gauge the model's performance. Commonly used metrics are Accuracy, Precision, Recall, F1 Score, Log loss, etc. It is worth noting that not all metrics can be used for all situations. For example, Accuracy cannot be used when dealing with imbalanced classification. Before diving deep into classification metrics, it is essential to know the Confusion Matrix in detail as it is the bedrock of most of the metrics that we will discuss.

Confusion Matrix

Confusion Matrix is an (n*n) matrix that measures the predictions of the classification model against the actual values. In the case of binary classification, the confusion matrix becomes a 2*2 matrix; the size of the matrix depends on the number of classes in the dependent variable. A typical Confusion matrix looks like below,

		Actual Values	
		1	0

Predicted Values	1	True Positive (TP)	False Positive (FP)
	0	False Negative (FN)	True Negative (TN)

Some of the terms mentioned in the above confusion matrix are defined as follows,

---

---

1. **True Positives**: When the actual class is positive and the model predicts a positive course, it is termed True Positive.

2. **True Negative**: When the actual class is negative, and the model predicts a negative type, it is True Negative.

3. **False Positive:** When the actual class is negative, and the model predicts a positive course, it is False Positive. One can think of it as the model falsely indicating a positive class when it is negative. False Positives are also known as **Type 1 errors**. For example, minimizing False Positives becomes essential in the Banking industry; if a customer is falsely predicted as a loan defaulter and the customer did not default, it is a loss of revenue to the bank.

4. **False Negative:** When the actual class is positive, and the model predicts a harmful category, it is False Negative. One can think of it as the model falsely predicting a negative course when the class is positive. False Negatives are also known as **Type 2 errors**. For example, minimizing the False Negatives becomes very important in the medical field; if a cancerous patient is diagnosed as non-cancerous, it can be fatal.

One should note that the aim of the build model should be to maximize the True Positives and True Negatives and minimize the False Positives and False Negatives. Now that we know the basic terminologies of a confusion matrix, we can look at the evaluation metrics derived from the confusion matrix.

Accuracy:

Accuracy is one of the most used metrics to evaluate model performance. It describes how accurate your model is. Mathematically, it is the ratio of the sum of True Positives and Negatives to the total number of data points. From the Confusion matrix, it can be derived as follows,

Accuracy = (TP + TN) / (TP+ TN+ FP+ FN)

There are a few things to note about Accuracy as an evaluation metric,

- Accuracy is a good metric when the classes in the dependent variable are balanced between positive and negative types.
- Accuracy is easy to calculate and easy to understand as well.
- High Accuracy in the case of imbalanced class distribution can lead to misleading results since the model might always predict the dominant class and might not predict the minor class.

Error Rate / Misclassification Rate:

Error rate or Misclassification rate is the exact opposite of Accuracy. It measures how inaccurate your model is. Mathematically, it is the ratio of the sum of False Positives and False Negatives to the total number of data points. It can also be calculated as 1-Accuracy.

Error Rate = (FP+FN) / (TP+TN+FP+FN) = 1 – Accuracy

True Positive Rate / Sensitivity / Recall:

Sensitivity measures how sensitive your model is. The model can correctly classify positive values. In simple terms, when the actual class is True or 1 or yes, how often does the model predict True or 1 or yes. Mathematically, it is the ratio of True Positives to Actual Positives. Sensitivity is an essential metric in the medical industry. If the model can predict a diseased individual as diseased, it is beneficial to the patient; the more correct predictions the model makes, the better it is.

Sensitivity = (TP) / (TP+FN)

False Positive Rate:

False Positive Rate measures the misclassifications. In simple terms, when the actual class is False or 0 or no, how often does the model predict True or 1 or yes. For example, if a patient is falsely expected as having a disease when he does not, it does not matter much as there are always False Positives that turn out in medical tests. Further tests can be conducted, and correct predictions can be obtained. Mathematically, FPR is the ratio of False Positives to the sum of True Negatives and False Positives.

FPR = (FP) / (FP+TN)

Factual Negative Rate / Specificity:

TNR or Specificity measures how specific our model is. If the model predicts all healthy individuals as not having a particular disease, the model is said to be highly specific. In simple terms, when it is No or 0 or False, how often does the model predict No or 0 or False. Mathematically, it is the ratio of True Negatives by the sum of True Negatives and False Positives.

Specificity = (TN) / (TN+FP)

Precision:

Precision measures how precise or accurate the prediction of your model is. In simple terms, when the model predicts True or Yes or 1, how often is the prediction correct? For example, when indicating fraudulent transactions, it is essential to predict trades as fraudulent correctly. If you expect non-fraudulent transactions as fraudulent, it can lead to business loss. Mathematically, it is the ratio of True Positives to the sum of True Positives and False Positives.

Precision = (TP) / (TP+FP)

F Beta Score / F1 Score:

F Beta score considers both precision and recall. There are instances where we need the model to e optimized for both precision and recall metrics. In such cases, the F Beta score is used as the metric. It is given by the equation below,

F Beta = (1+Beta^2) * ((Precision*Recall) / (Beta^2*Precision + Recall))

Another vital evaluation metric is the F1 Score. We all know it as the Harmonic mean of precision and recall metrics, and it is derived from the above equation by substituting Beta = 1. When we substitute Beta with 1, we give equal importance to both Precision and Recall metrics.

F1 Score = (2*Precision*Recall) / (Precision + Recall)

Another essential thing to note about the F1 Score is that it depends on TPR and FPR now, and these values can be altered by altering the threshold of the classifier. For example, for the default threshold of 0.5, there are specific TPR and FPR; if you alter the threshold value, the TPR and FPR change and hence the value of the F1 Score changes.

Log loss:

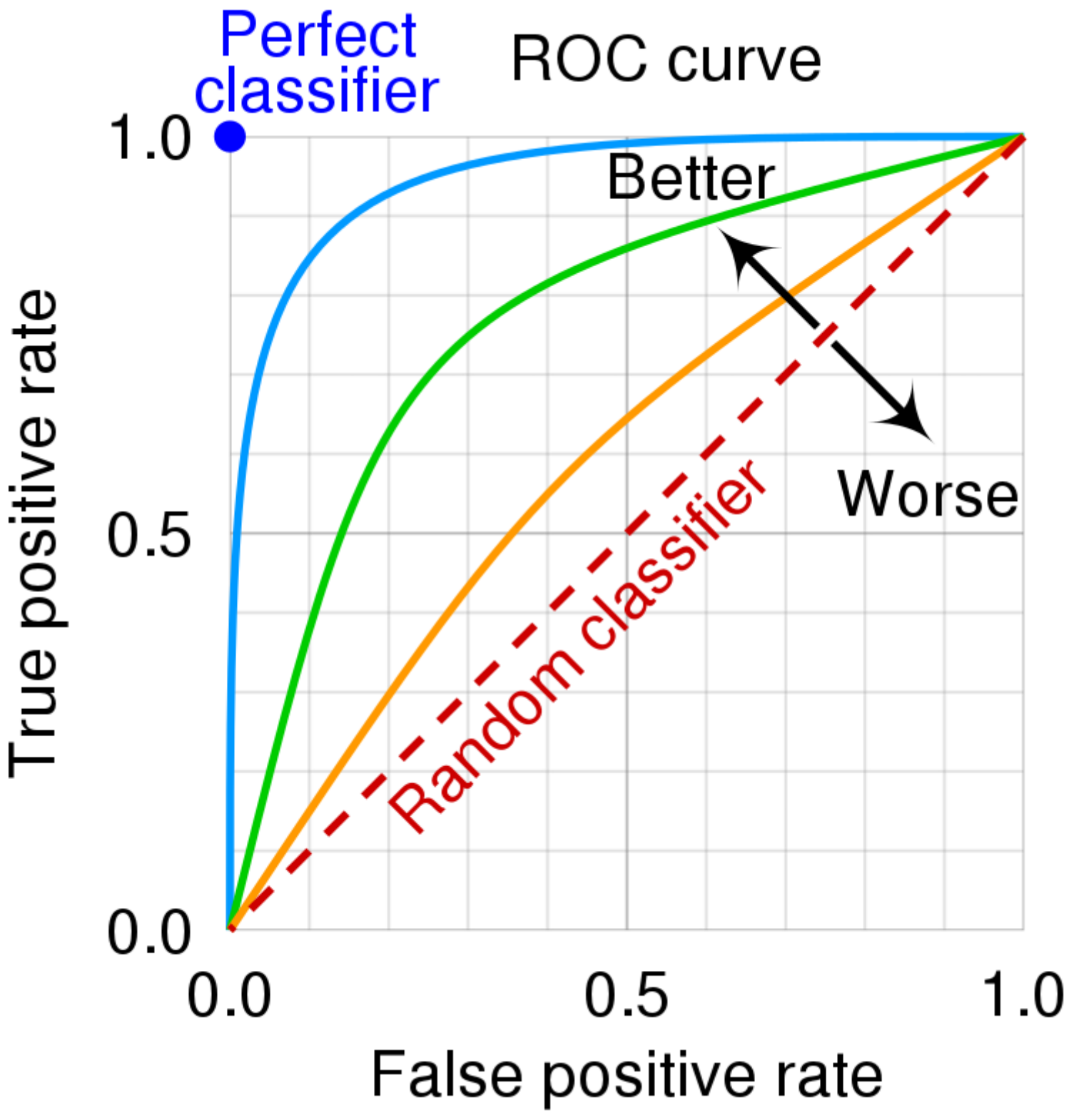
Log loss is a vital evaluation metric used to compare the performance of two classification models. Lower the log loss, better is the model in short. Log loss penalizes the false classifications. ***If the model assigns a lower probability to the correct class for a particular data point, then the log loss of the corresponding data point will be significantly significant. Similarly, if the model gives a higher probability to the incorrect class, the log loss will be higher***. So basically, the higher is the probability assigned to the correct class, the lower is the log loss. Log loss for a binary classification problem is given by the formula shown below,

$$-\frac{1}{N}\sum_{i=1}^{N}[y_i \log, p_i + (1-y_i) \log, (1-p_i)].$$

Source: https://www.analyticsvidhya.com/blog/2020/11/binary-cross-entropy-aka-log-loss-the-cost-function-used-in-logistic-regression/

ROC Curve / Area Under the ROC Curve (ROC AUC Score):

Before diving into what the ROC curve is, it is essential to note that most of the Machine Learning models do not predict the class labels directly; they consistently indicate the probability of a data point belonging to either positive or negative class, then based on the threshold value (which is .5 by default), the labels are classified as belonging to either positive or negative type. ROC Curve stands for Receiver Operating Characteristic Curve. ***The curve checks how the observations change classes based on the variations in the threshold value; it visualizes the tradeoff between TPR and FPR rates.*** It is a graph of True Positive Rate or Sensitivity on the Y-axis and False Positive Rate or (1-Specificity) on the X-axis. It is worth noting that if your model is better, the curve hugs the Y-axis as much as possible. A typical ROC curve looks like the figure below,



Source: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

One can also note that practically, the curves are not as perfect as shown in the figure above. Likely, they will be very different from the above figure.

One more critical metric that can be calculated from the ROC curve is the Area Under the ROC Curve; the higher the AUC score value, the better the model performance. If the ROC curve hugs the Y-axis or is closer to the Y-axis, the AUC score will be higher. The maximum possible value of the AUC score is 1, which is practically not possible as the model cannot predict all observations correctly.

Regression Metrics:

We saw how to evaluate the performance of a classifier till now. We will now deep dive into evaluating the performance of a Regression model where we predict continuous values and not individual classes. The Regression Evaluation metrics differ from classification evaluation metrics, and the most popular ones are MAE, MSE, RMSE, R Squared, etc.

Mean Absolute Error:

The term Error in "Mean Absolute Error" stands for the difference between the actual and the predicted values of the continuous variable. When we predict a constant variable, some of the values predicted can be below the actual value, and some can be above the actual value. If you consider the sum of the differences between actual and predicted, some values may cancel out, which is why we take the absolute value of the difference between actual and predicted. This cancels out any negative values, and it is the average fundamental value of the differences between actual and predicted values.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Some important points to note about the MAE as an evaluation metric are,

- The MAE is in the same units as the continuous dependent variable.
- It is easy to interpret MAE.
- Although it takes the absolute value of the differences, it does not highlight the extreme values of the differences, i.e., it does not highlight predictions that are way off the accepted range.
- It is not sensitive to outliers.

Mean Squared Error:

Like the MAE metric, MSE measures the differences between the actual values and the predictions. It takes the square of the differences between accurate predictions instead of the absolute values. Mathematically, it is the average squared differences between actual and predictions.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Some important points to note about MSE are,

- MSE is sensitive to outliers as we take the square of the differences, which highlights extreme values.
- It is not easy to interpret as it does not have the same units as the continuous dependent variable.
- It is better than MAE.

Root Mean Squared Error:

RMSE is almost the same as MSE, except it takes the square root of the Mean Squared Error. It is the most popular evaluation metric, and it overcomes any drawbacks that MAE or MSE have. Mathematically it is the square root of the average of the squared difference between actual values and predictions.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Some important points about RMSE are,

- It is sensitive to outliers as it takes the squared differences.
- It has the same units as that of the continuous dependent variable.
- It is easier to interpret as compared to MSE.

Root Mean Squared Log Error:

RMSLE is almost the same as RMSE except that it takes the log values of the actual and predicted values instead of using them as is. It also adds 1 to the weights if the value is 0 as the log of 0 is not defined. It is also not valid as a metric when negative values are involved.

Root Mean Squared Log Error (RMSLE)

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

Some important points to note about RMSLE are,

- It is used when the target variable has an extensive range.
- It is also used when we look at the growth over the years or when there is exponential growth.
- It is also used to know the percentage error as the expression within evaluated as a ratio as well (applying logarithmic rules).

MAPE:

MAPE stands for Mean Absolute Percentage Error. It can be interpreted as the average of the absolute percentage of errors. It is mainly used as an evaluation metric in forecasting problems where we need to determine the values in the future. It is calculated as the absolute value of the ratio of the difference between actuals and predicted to the actual values. It is given by the formula below,

$$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

Some of the critical points to note about MAPE are,

- It cannot be used when the actual values are 0 because you cannot divide something by zero.
- A good MAPE value is always less than 10%.

R Squared:

R Squared is unlike the regression metric that we discussed above. It considers the predictions of our model and the average value of our dependent variable. There are two ways in which R Squared can be interpreted; one ***is that it describes how much variation in the dependent variable is explained by our current set of independent variables,*** the other interpretation is that ***it tells how better our current model is as compared to just predicting the average value of the dependent variable for all data points.*** R Squared is defined by the formula given below,

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

As we can see above, in the numerator of the formula, we subtract the predicted values of our model from the actual values and square them. In the denominator, we remove the average value of our dependent value from the fundamental values and square it.

Some of the essential points to note about R Squared are,

- The higher the R Squared, the better the model (although the Adjusted R contests this Squared metric below).
- As we keep adding variables into the model, R Squared increases, and it does not decrease. As we add more variables, although the denominator remains the same, the numerator reduces, and hence the R squared keeps increasing.
- R Squared ranges between 0 and 1.

Adjusted R Squared:

Adjusted R Squared is the modified version of R Squared. It considers the number of data points, the number of predictors in the model, the R Squared value. Adjusted R Squared is a more reliable metric than R Squared because it mainly considers the number of predictors. ***If we keep adding variables to our model that do not contribute to the model's performance, the Adjusted R Squared does not increase; only if the variable added is contributing to improving the model, the Adjusted R Squared value increase.***

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

Some important points about Adjusted R Squared are,

- Adjusted R Squared might decrease, unlike R Squared.
- Adjusted R Squared is less than or equal to R Squared.
- It penalizes for adding more insignificant variables to our model.

End Notes:

Although many other evaluation metrics are used for classification and regression problems, the ones explained in the article are the most used.

Python code to implement all the metrics mentioned in the article can be found in the following GitHub link: here.

I am a Software Test Engineer with a passion for Data Science; I am looking to explore opportunities in Data Science and Machine Learning; you can connect with me on GitHub and LinkedIn.

Check out my articles on Data Scraping here and Odds Ratios here.

As always, any suggestions tips are always welcome.

blogathon evaluation metrics machine learning python

About the Author

Vishwanath Kulkarni

Our Top Authors

view more

Download

Analytics Vidhya App for the Latest blog/Article

Previous Post

Virtual Zoom using OpenCV

Next Post

Using Sequential Model to Predict Prices of Real Estate

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

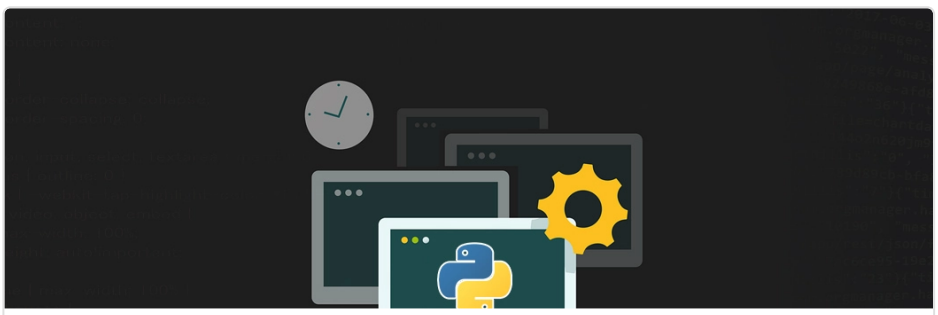
Website

Notify me of follow-up comments by email.

Notify me of new posts by email.

Submit

Top Resources



Python Tutorial: Working with CSV file for Data Science

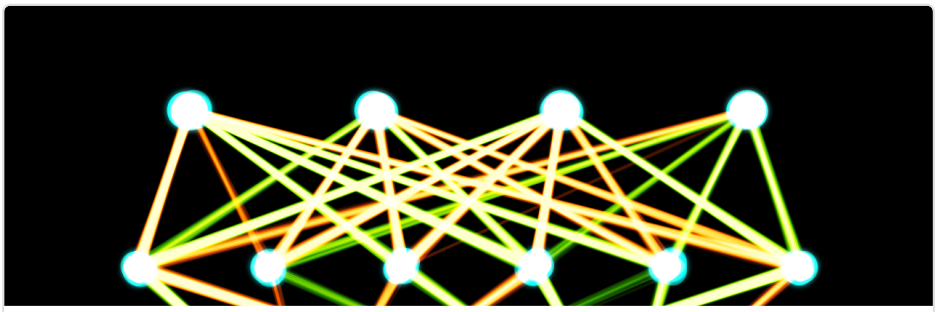
Harika Bonthu - AUG 21, 2021

30 Questions to test a data scientist on Linear Regression..

1201904 - JUL 03, 2017

30 Questions to test your understanding of Logistic Regression

1201904 - AUG 03, 2017



Boost Model Accuracy of Imbalanced COVID-19 Mortality Prediction Using GAN-based..

Bala Gangadhar Thilak Adiboina - OCT 07, 2020

Analytics Vidhya

About Us

Our Team

Careers

Data Scientists

Blog

Hackathon

Discussions