**ONE HOT ENCODING**

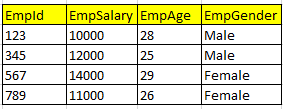
Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric.

In general, this is mostly a constraint of the efficient implementation of machine learning algorithms rather than hard limitations on the algorithms themselves.

This means that categorical data must be converted to a numerical form. If the categorical variable is an output variable, you may also want to convert predictions by the model back into a categorical form in order to present them or use them in some application.

**pandas.get\_dummies()** is used for data manipulation. It converts categorical data into dummy or indicator variables.

Suppose:



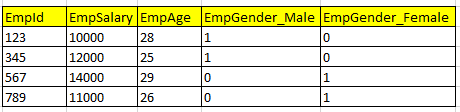
If you feed this training data to regression model in Python, then what internally happens is python’s regression library will try to fit an equation of the form

***y = m1x1+m2x2+m3x3+c***

to the given data. Here x1,x2 etc are features of the training data and y is response variable.

Above equation makes sense till all the x’s are continuous numbers or real numbers, however if one of the x is categorical in nature, for example **gender**in our example, the equation become meaningless. (***2\*Female or 3\*Male does not make any sense)***

Hence before feeding above data to regression models, we need to encode these categorical variable to real values.

After getting dummies:  


These new features in the data(***EmpGender\_Male,EmpGender\_Female***) are called ***dummy variables***.

In one hot encoding, a new binary variable is added for each unique integer value.

In the “color” variable example, there are 3 categories and therefore 3 binary variables are needed. A “1” value is placed in the binary variable for the color and “0” values for the other colors.

For example,

|  |  |
| --- | --- |
| 1  2  3  4 | red, green, blue  1, 0, 0  0, 1, 0  0, 0, 1 |

The binary variables are often called “dummy variables” in other fields, such as statistics.

**CONFUSION MATRIX**

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known.

If the data is about whether a person has disease or not

* **true positives (TP):** These are cases in which we predicted yes (they have the disease), and they do have the disease.
* **true negatives (TN):** We predicted no, and they don't have the disease.
* **false positives (FP):** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
* **false negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

**Accuracy:** Overall, how often is the classifier correct?

* (TP+TN)/total = (100+50)/165 = 0.91

**Precision:** When it predicts yes, how often is it correct?

* + TP/predicted yes = 100/110 = 0.91

**Recall**: Recall tells us how many of the actual positive cases we were able to predict correctly with our model.

**F Score:** This is a weighted average of the true positive rate (recall) and precision.

**F1-score is a harmonic mean of Precision and Recall.** The **Fbeta-measure** is a generalization of the F-measure that adds a configuration parameter called beta. A default beta value is 1.0, which is the same as the F-measure. A smaller beta value, such as 0.5, gives more weight to precision and less to recall, whereas a larger beta value, such as 2.0, gives less weight to precision and more weight to recall in the calculation of the score.

It is a useful metric to use when both precision and recall are important but slightly more attention is needed on one or the other, such as when false negatives are more important than false positives, or the reverse.

**ROC Curve:** This is a commonly used graph that summarizes the performance of a classifier over all possible thresholds. It is generated by plotting the True Positive Rate (y-axis) against the False Positive Rate (x-axis) as you vary the threshold for assigning observations to a given class.

**PREDICTION AND MODELS**

**X\_train** - This includes your all independent variables,these will be used to train the model, also as we have specified the test\_size = 0.4, this means 60% of observations from your complete data will be used to train/fit the model and rest 40% will be used to test the model.

**2). X\_test** - This is remaining 40% portion of the independent variables from the data which will not be used in the training phase and will be used to make predictions to test the accuracy of the model.

**3). y\_train** - This is your dependent variable which needs to be predicted by this model, this includes category labels against your independent variables, we need to specify our dependent variable while training/fitting the model.

**4). y\_test** - This data has category labels for your test data, these labels will be used to test the accuracy between actual and predicted categories.

***Training Score:*** How the model generalized or fitted in the training data. If the model fits so well in a data with lots of variance then this causes over-fitting. This causes poor result on Test Score. Because the model curved a lot to fit the training data and generalized very poorly. So, generalization is the goal.

***Validation Score*** This is still a experimental part. We keep exploring our model with this data-set. Our model is yet to call the final model in this phase. We keep changing our model until we are satisfied with the validation score we get.

***Test Score*** This is when our model is ready. Before this step we have not touched this data-set. So, this represents real life scenario. Higher the score, better the model generalized.

The classification report visualizer displays the precision, recall, F1, and support scores for the model. In order to support easier interpretation and problem detection, the report integrates numerical scores with a color-coded heatmap. All heatmaps are in the range (0.0, 1.0) to facilitate easy comparison of classification models across different classification reports

Models: (Harsh)

**ROC CURVE**

An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

* True Positive Rate
* False Positive Rate

**True Positive Rate** (**TPR**) is a synonym for recall and is therefore defined as follows:

TPR=TP/(TP+FN)

**False Positive Rate** (**FPR**) is defined as follows:

FPR=FP/(FP+TN)

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.

**K CROSS VALIDATION**

K-Folds technique is a popular and easy to understand, it generally results in a less biased model compare to other methods. Because it ensures that every observation from the original dataset has the chance of appearing in training and test set. This is one among the best approach if we have a limited input data. This method follows the below steps.

1. Split the entire data randomly into K folds (value of K shouldn’t be too small or too high, ideally we choose 5 to 10 depending on the data size). The higher value of K leads to less biased model (but large variance might lead to over-fit), where as the lower value of K is similar to the train-test split approach we saw before.
2. Then fit the model using the K-1 (K minus 1) folds and validate the model using the remaining Kth fold. Note down the scores/errors.
3. Repeat this process until every K-fold serve as the test set. Then take the average of your recorded scores. That will be the performance metric for the model.

**FEATURES IMPORTANCE**

**Feature importance** refers to techniques that assign a score to input features based on how useful they are at predicting a target variable.

There are many types and sources of feature importance scores, although popular examples include statistical correlation scores, coefficients calculated as part of linear models, decision trees, and permutation importance scores.

Feature importance scores play an important role in a predictive modelling project, including providing insight into the data, insight into the model, and the basis for [dimensionality reduction](https://machinelearningmastery.com/dimensionality-reduction-for-machine-learning/) and [feature selection](https://machinelearningmastery.com/rfe-feature-selection-in-python/) that can improve the efficiency and effectiveness of a predictive model on the problem.