**Machine Learning Notes**

Data Preprocessing

1. Libraries –

-NumPy will allow us to work with arrays.

-Matplotlib will allow us to plot charts and graphs.

-Pandas (super useful library to preprocess the dataset) will allow us to import the dataset and allow us to create matrix of features and the dependent vector.

1. Importing the Dataset –

The Pandas function that we use to read the dataset will read all the values from the source file and create a data frame, that will contain the exact rows and columns as that of the source file.

The first step would be to import the dataset.

In the next step we create two new entities - Matrix of Features, and Dependent Variable Vector

The above-mentioned entities are universal to every dataset that you use.

Features – They are the column with which we shall predict the dependent variables.

Dependent Variables – The columns on which we will run the machine learning algorithms to predict the future values based on the information given in the feature columns.

So basically, the features/independent variables are the variables containing some information which will be used to predict the dependent variables.

Usually, not always, the dependent variables are stored in the last column of your dataset.

So, we will create two separate variables – one to store the independent variables (array), one to store the dependent variables (vector).

1. Taking Care of Missing Data –
2. Ignore the missing data by deleting it. This method works when you have a large dataset. So, let’s say you only have 1% of missing data, then it will be okay to delete the missing data as it isn’t much.
3. Replace the missing data by the average of all the values of the column in which the data resides. It is the classic way of handling missing data.
4. We use skitlearn library to handle the missing data. This library is one of the most important libraries when it comes to machine learning, as it will be used to create various models used in machine learning.
5. We will use the class simple imputer class from skitlearn to handle missing data.
6. First, from sklearn.impute, we import SimpleImputer
7. Then, we will import the simple imputer class, then create an instance/object of the class. And that object will allow us to handle the missing data.
8. Syntax = imputer = SimpleImputer(missing\_values = np.nan, strategy = 'mean')
9. Remember, we have only created an object for the imputer class, and we haven’t connected anything yet to our matrix of features. The next step would be to apply the object on our Matrix of Features (Independent Variable Array).
10. Note, a class contains an assemble of instructions, and some operations and actions that you can apply to other objects and variables, and those operations and actions are nothing but methods (they are like functions).
11. We will use fit() method. It will connect the imputer to matrix of features. It will look at the missing value columns and count the average of that column.
12. And to replace the missing values we will use the transform() method.
13. Encoding Categorical Data –

Encoding The Independent Variable -

1. Every dataset has categorical data column, like how we had the country column in the one we used to understand preprocessing. So, it will be difficult for the machine learning algorithm to calculate correlations between the categorical variable (features) column and dependent variable columns.
2. Therefore, we will have to turn the strings/categories into numbers.
3. If we encode the data as 0, 1, 2 and so on, then our future machine learning model could understand that there is a numerical order between those categories, and it will interpret that the order matters, when it doesn’t. This will in turn result into some misinterpreted correlations between features and dependent variables.
4. And to avoid that we perform “One Hot Encoding”.
5. One Hot Encoding will turn the single categorical column into as many columns as the classes/categories of the column.
6. It also consists of creating binary vectors for each category, ensuring that there is no numerical order among the categories.
7. E.g. – France = 100, Spain = 010, Germany = 001.
8. We can replace the dependent variables with 0’s and 1’s, if it is a binary outcome (purchased, not purchases). Doing so won’t compromise the future accuracy of the model.
9. To implement one hot encoding, we use two classes – 1. Column transformer class from compose module of skit learn library and 2. One Hot Encoder Class from preprocessing of SkitLearn library.
10. We define two things when we create an object of ColumnTransformer Class – transformers (we specify what kind of transformation we want to do, what kind of encoding would we like to perform and on which indexes of columns we want to transform), and remainder (specifies the columns that won’t be applied with any transformation)
11. After defining that we can directly call the function on the Matrix of Features, using the fit transform function. The fit\_transform function is a combination of fit and transform function – it does the calculation part of the columns and updates the column.
12. Since the transformed array won’t be a NumPy array, and the future machine learning model expects it to be a NumPy array, we will convert the encoded array into a regular array while calling fit\_transform().

Encoding The Dependent Variable –

We will use the label encoder class from skitlearn library to encode the dependent variable column. We can use LabelEncoder class when we have two classes in a categorical variable column which will convert the class data into binary data.

1. Splitting The Dataset into Training Set and Test Set –
2. We must apply feature scaling after splitting the data.
3. Test is supposed to be like a brand-new set on which you apply your machine learning model to test for its accuracy. And as feature scaling takes the mean of the whole set to scale the data, if we apply feature scaling before the split, it will use the mean of the whole dataset and scale the data according to that – including the test set data. And doing that will result in data leak on test set.
4. We don’t apply feature scaling before the split to avoid data leak on the test set.
5. We use a module by sklearn called “Model Selection”, which contains a function called “train, test, split”.
6. We will get 4 sets of data – training set for matrix of features and predictor variables, and a test set for the same.
7. Syntax - from sklearn.model\_selection import train\_test\_split
8. Recommended size of split – 80% in training set and 20% in test set.
9. random\_state = 1 is nothing but the same as set\_seed() in R.
10. Feature Scaling –
11. Feature Scaling will allow us to put all our feature (independent) variable to the same scale.
12. We need to do this to avoid some features to be dominated by other features.
13. For instance, age will be dominated by salary if we don’t feature scale the variables.
14. We won’t need to apply feature scaling for all machine learning models.
15. For example – in multiple linear regression, each variable is multiplied by a coefficient in the linear regression equation. So, if we have variables taking higher values, the coefficients will compensate by taking small values for variables with high values.
16. We apply feature scaling to the dependent and independent variables if the independent variable is not binary. Sometimes the independent variable could be something like salary, which would be beyond the range of feature scaling. For example – when looking at some database where we predict whether a customer will purchase something or not, we encode the purchase decision as categorical variable and it results it 0 and 1, which is in the range of the feature scaled results. And thus, we don’t need to feature scale the independent variable. But sometimes, the dependent variable could be something like salary, which needs to be feature scaled, as it is beyond the range of feature scaled values of the independent variables.
17. If we are applying feature scaling of two set of variables, we cannot use the same scaler object for both the set of variables, because when we apply the scaler object on independent variable set, it will compute the mean and the standard deviation of that variable and because we don’t have same mean and same standard deviation for the other set of variables, we create two standard scaler objects.

Graphical user interface, application

Description automatically generated

1. The above given are the main two techniques for feature scaling.
2. Standardization is good for normally distributed variables, as it keeps the normal distribution but converts it to ensure zero mean and unit standard deviation.
3. Whereas normalization is useful when we don’t know how the data is distributed.
4. We use sklearn class called “StandardScaler” which will perform standardization on our datasets.
5. We don’t apply feature scaling on dummy variables in the Matrix of Features.
6. Standardization transforms your features in such a way that your variables take values between -3 and +3.
7. And if the feature scaling is applied to the dummy variables, it will result into non-sense values, and we won’t be able to identify which dummy variable represents what.

(Dummy Variables are nothing but the encoded categorical variables)

1. And scaling of dummy variable won’t result into a better and more accurate model.
2. When we transform the test set, we only apply the transform method, as it needs to be scaled by the same scaler used on the training set, we cannot use a new scaler.
3. If we apply fit\_transform method on the test set, we shall end up with a new scaler, and that would not make sense because the test set will be the input of the predict function that will return the predictions after the model has been trained. And since the machine learning model will be trained with a particular scaler (the scale applied on the training set), we will need that scaler to make predictions that will be congruent with the way the model was trained.