General steps for model creation:

* Explore a clean the data
* Split data in trin/validate/test module
* Fit initial data to form model and evaluate
* Tune hyperparameter (find a good single of group of parameters)
* Test and validate the set and the final model evaluated.

The dataset taken is Titanic dataset:

Aim is to find those people who are most likely to be alive after the incident using the data available in the dataset.

In the dataset the predicting column is ‘survived’ after the crash. Rest column can be used to determine the value for predicting column.

Cleaning Process:

Firstly, load the dataset, then find all missing values in the dataset and fill them with appropiate values (for eg. Filling the missing age values with avg or mean values on the rest present age values)

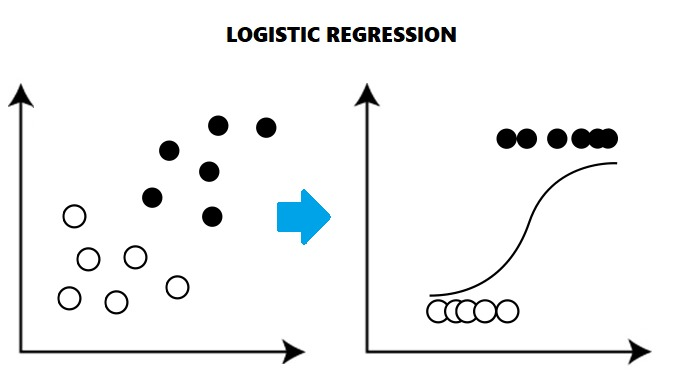
After doing the categorial plot of the data it was defined that the survival rate is max for people with siblings around 2 and parent children 2-3.

Regression: A statistical process for estimating relations between different variables and to predict certain outcomes. E.g. Linear Regression is to use single line equation to understand relation between 2 variable s or 2 set of related variables and by determining the linear model that can nearly fit maximum possible value pairs so that can predict one for another often y-axis variable over given x-axis variable values.

Logistic regression: A form of regression where the output is often the output is binary. A linear regression can’t be used when the unit is binary as the points will on specifically 2 Y values and fitting a line over it will be impossible and it will end up predicting values below 0 or over 1 which is unacceptable and hence logistic regression comes into the picture.



Logistic regression doesn’t do well with messy data and massive unit of data, along with when the data has continuous target variables (target values should not have values greater than 1 or well below 0). It also often gives wild outputs when the data frame is too thin (few columns and too many rows) and too short (too many columns but few rows).

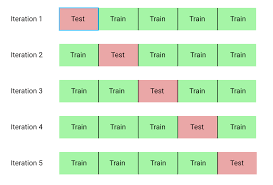


To work on regression, you must understand about overfitting and regularization.

Overfitting: The situation when the model touches every point given in training data. This is bad for prediction as it often gives worst success rate in validation and behaves badly while testing.

Regularization: The technique used to reduce overfitting by shrinking the coefficient estimate towards zero.

Although, as per the equation C is considered regularization variable but in reality is the regularization variable. The lower is C the high regularization and complexity will be low in the model and Model so likely to underfit but as C tends to infinity the regularization will be very high and the model will also by very complex and likely to overfit.



So, it is very difficult to find C and very crucial to find the value of C to make the model fit properly over the data. Here to find the value of C, we do K-fold cross validation.

K-Fold cross validation: In this the model will split the dataset in k groups and will iterate it k times. It will fit the model to k-1 subset and then test it in subsequent subsets.

Support Vector Machines (SVM): A classifier that finds an optimal hyperplane that maximizes the margin between two classes. A being a classification technique, it works best with text data but not a massive amount of data. To get a correct SVM (Support Vector Machines) model means to get a hyperplane that differentiates all possible classes with largest possible margins between the hyperplane and classes region. The term support vector comes from a perpendicular line that can be drown from the optimalhyperplane and the line in class region where first point occurs, and the goal is to make that line a maximum as possible.

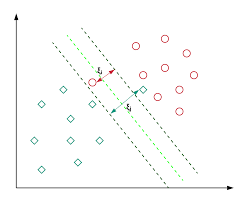
Kernel Trick: this trick is very important while working on data using SVM as in this trick if the data can’t be separable by linear plane in n-dimension then it will convert the data into n+m dimension where it can be separable. E.g. If a dataset is not separated in 3-d dimension so it will gradually increase the dimension till it finds that dimension where it can be separated using single linear plane.

SVM can be used on continuous and categorial dataset but not good with regression. Also, when excel file has a smaller number of rows and high number of features, the complexity of the problem is very high, or data has different categorial points co-inside often to each other SVM can be used.

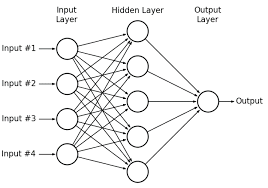
On the downside SVM is very slow and takes a lot of time to train and validate, As the dimensions increases its very difficult to find out what is happening inside SVM so if output is needed every time after every step then SVM is not your thing.

SVC(C=1.0, break\_ties=False, cache\_size=200, class\_weight=None, coef0=0.0, decision\_function\_shape='ovr', degree=3, gamma='scale', kernel='rbf', max\_iter=-1, probability=False, random\_state=None, shrinking=True, tol=0.001, verbose=False)

This is how an SVC or Support Vector Classifier looks like while calling. As we have only discussed about 1 concept that is kernel Trick which as we can see is our hyper parameter here and need our attention. Along with kernel we obviously going to investigate C, as discussed during Logistic Regression C defines regularity and is important to find exact value to converge to get the best prediction for label. In the case of SVM, high C means very small margin hence tends to overfit (The classification plane will be too close to all points and hence very difficult to converge) and C to 0 will be too much margin and might be underfit.



Multilayer Perceptron: A connected series of nodes that kind of form acyclic graph where the data it is processed through these nodes namely called as function or a model. It is a type of neural network where each of these nodes get input data and act like an individual logistic Regression.



In multi-layer perception, just before the output layer there are pseudo hidden output layers which predicts the outcome for each hidden layer taking input and then the main output layer is the cumulative output defining ho w the prediction should go.

When want to do prediction in a very strict, controlled environment and just caring about the performance then multi-layer perception is a good tool. Can do categorial and binary target variables and tend to have very complex relationship due to multiple hidden layers. But it is still bad with working on images and time series problems. Better use CNN (Convolution Neural Networks) for these categories of projects.

The MLPRegressor is of Regression problems:

MLPRegressor(activation='relu', alpha=0.0001, batch\_size='auto', beta\_1=0.9, beta\_2=0.999, early\_stopping=False, epsilon=1e-08, hidden\_layer\_sizes=(100,), learning\_rate='constant', learning\_rate\_init=0.001, max\_fun=15000, max\_iter=200, momentum=0.9, n\_iter\_no\_change=10, nesterovs\_momentum=True, power\_t=0.5, random\_state=None, shuffle=True, solver='adam', tol=0.0001, validation\_fraction=0.1, verbose=False, warm\_start=False)

For Classification problem:

MLPClassifier(activation='relu', alpha=0.0001, batch\_size='auto', beta\_1=0.9, beta\_2=0.999, early\_stopping=False, epsilon=1e-08, hidden\_layer\_sizes=(100,), c='constant', learning\_rate\_init=0.001, max\_fun=15000, max\_iter=200, momentum=0.9, n\_iter\_no\_change=10, nesterovs\_momentum=True, power\_t=0.5, random\_state=None, shuffle=True, solver='adam', tol=0.0001, validation\_fraction=0.1, verbose=False, warm\_start=False)

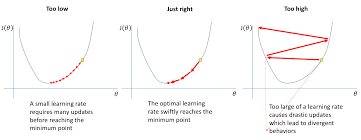
There 3 hyperparameters which are needed more attention which will require much better understanding of it. And these 3 hyperparameters are:

Activation: Activation function defines what kind of function or equation needed in hidden layer. Following are activation functions that are often used: Sigmoid, ReLU, TanH.

* Sigmoid function: Kind of same as Logistic Regression often used of target variables with major 2 output values for the target variable with somewhat same formula as logistic regression.
* TanH Function: As the name suggest formula came with Tan as basic functoin fo the prediction target variable. It is always between –1 to 1 for target value.
* ReLU function: abbrevation of Rectified Linear Unit, has decision formula for target variable with 0 till some values and diagonal plane/ Line for rest of x-values henceforth.

Hidden\_layer\_sizes: It defines the number of hidden layers and number of nodes that are there going to be in each hidden layer.

learning\_rate: Learning rate is often associated with finding global minima (not local) as soon as possible with adequate resources possible. Getting as soon as possible doesn’t mean keeping learning\_rate high, as if you keep it high then it will just jump on certain points and end up missing minima on every turn. On the other hand, keeping it too low will achieve the first local minima (we need global minima for prediction) but will end up consuming a lot of resource power and will left with too less to do the prediction.



For the learning\_rate there are 2 values to choose between:

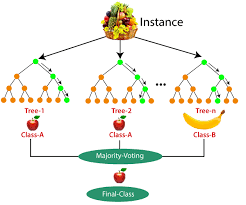
* ‘Constant’: as name suggest the learning\_rate will be constant and will not change even if the jump comes nearer to minima.
* ‘invscaling’: The learning\_rate decreases with every jump and allows to reach minima little early.
* ‘adaptive’: initially rate will be constant but as it reaches near minima it starts to decrease and increases precision.

Random Forest

It’s a collection of multiple decision trees working independently and providing more accurate and stable prediction when converged together. It’s a type of ensemble model which is combination of more than one model more independent and un-biased result with minimum variance for more accurate and fair result.

In random forest, there will be n subsets of redundant values and subset are done in terms of both rows and columns. Each subset will create their own decision tree and each decision tree does not have any idea about other decision trees information. For testing, a sample dataset will be processed by each decision tree and the answer will be compared and majority voting will be done and then it will be result.

It is good for both categorial and continuous target values, it gives significance for relationship between every major predictor. Even if data is too messy or have missing values then this can be used for quick result.



RandomForestClassifier(bootstrap=True, ccp\_alpha=0.0, class\_weight=None, criterion='gini', max\_depth=None, max\_features='auto', max\_leaf\_nodes=None, max\_samples=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0, n\_estimators=100, n\_jobs=None, oob\_score=False, random\_state=None, verbose=0, warm\_start=False)

RandomForestRegressor(bootstrap=True, ccp\_alpha=0.0, criterion='mse', max\_depth=None, max\_features='auto', max\_leaf\_nodes=None, max\_samples=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0, n\_estimators=100, n\_jobs=None, oob\_score=False, random\_state=None, verbose=0, warm\_start=False)

As the dataset we are using requires classifier function, we will focus on main 2 following parameter from the given parameters in the classifier:

* n\_estimators: This to define maximum number of trees will be there in the model to work, and
* max\_depth: This to maximum length that a tree can cover at maximum.

Procedure for model creation for all above given models:

**Importing joblib for saving the model**

Import joblib

**Read csv files for data reading**

Import pandas as pd

feature = pd.read\_csv('././Titanic\_x\_train.csv')

label= pd.read\_csv('././Titanic\_y\_train.csv',header=None) #added header so that the system does not take first column as header.

**Import the classifier and GridsearchCV for cross fold validation**

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import GridSearchCV

**Define classifier as function, parameter and cross fold function**

rfc = RandomForestClassifier()

parameter = {

'n\_estimators' : [5,50,100,250],

'max\_depth' : [2,4,8,16,32, None]

}

cv = GridSearchCV(rfc, parameter, cv = 5)

**Calling fit function for cv parameter**

cv.fit(feature, label.values.ravel())

**To print best possible pair of parameters for model**

def ML\_results(results):

mean = results.cv\_results\_['mean\_test\_score']

std = results.cv\_results\_['std\_test\_score']

for mean, std, params in zip(mean, std, results.cv\_results\_['params']):

print('mean: ',round(mean,3),' std: ',round(std \* 2,3),' for ',format(params))

print('\n Final parameter decided: ',format(results.best\_params\_))

ML\_results(cv)

Boosting:

Nearly same as random forest but it is combined form of weak models to create a big strong model. It takes a weak model (with less than 50% prediction right) and trains it, then rerun that model but with stronger way by learning last one mistake and do this n times to make it more strong and accurate.

For this dataset we will be using gradient boosting which is one of all possible boosting available.

GradientBoostingClassifier (ccp\_alpha=0.0, criterion='friedman\_mse', init=None, learning\_rate=0.1, loss='deviance', max\_depth=3, max\_features=None, max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0, n\_estimators=100, n\_iter\_no\_change=None, presort='deprecated', random\_state=None, subsample=1.0, tol=0.0001, validation\_fraction=0.1, verbose=0, warm\_start=False)

GradientBoostingRegressor (alpha=0.9, ccp\_alpha=0.0, criterion='friedman\_mse', init=None, learning\_rate=0.1, loss='ls', max\_depth=3, max\_features=None, max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0, n\_estimators=100, n\_iter\_no\_change=None, presort='deprecated', random\_state=None, subsample=1.0, tol=0.0001, validation\_fraction=0.1, verbose=0, warm\_start=False)

From the given parameters we must investigate 3 parameters:

* learning\_rate
* n\_estimators
* max\_depth

Apart from these 5 most used models which to use for what kind of problem, this is very difficult question. To answer these questions, we need to answer set of questions on 2 major criteria - Accuracy and Latency.

The questions are like:

Latency-

How fast it will train? - faster training go for logistic slower goes for random and boosting

How fast it will predict? - fast and real time prediction go for gradient boosting, for slower can go for randomforest.

How the data is in size:

Short and fat (lot of features but low amount of data): SVM is much faster,

Long and skinny (low feature but a lot of data): Same for SVM

Testin and evaluation is done in .py file please look into that for understanding