Linear Regression

what is supervised machine learning algorithms?

It is a type of machine learning where the algorithm learns from labeled data. Labeled data means the dataset whose respective target value is already known. Supervised learning has two types:

- **Classification**: It predicts **the class of the dataset** based on the independent input variable. Class is the categorical or discrete values. like the image of an animal is a cat or dog?
- **Regression**: It predicts **the continuous output variables** based on the independent input variable. like the prediction of house prices based on different parameters like house age, distance from the main road, location, area, etc.

For Sklearn users:

Rules for Regression:

1. Features and label must be in the form of numpy array
2. Features must be in 2d array
3. Label must be in 2d array

ML coding begins

Before you initiate the coding, you must know two things from your data scientists:

1. Approved **Significance level** for the project
2. Timeline to develop and deploy the model

1. Create Train Test Split
2. Build the model
3. Check the Quality of the Model
4. If Satisfied, perform Deployment; else go to step 2

Rule check the quality of the model:

The best way to check the quality of the model is:

1. Ensure your test score > train score (Model must be perform best on UNKNOWN DATA !!!)
2. Ensure your test score >= (1 - SL)

1		`salaryData = pd.read_csv('Salary_Data.csv')` `salaryData.head()` **YearsExperience** **Salary** 0 1.1 39343.0 1 1.3 46205.0 2 1.5 37731.0 3 2.0 43525.0 4 2.2 39891.0
2	Seperate data as features and label	`features = salaryData.iloc[:,[0]].values` `label = salaryData.Salary.values.reshape(-1,1)`
3	1- Create Train Test Split Random_State is like seeds and we use it to create samples to see which one is better (we Call it Data Randomizations)	`from sklearn.model_selection import train_test_split` `X_train,X_test,y_train,y_test = train_test_split(features,` `label,` `test_size=0.2,` `random_state=10)`
4	2- Build the model	`from sklearn.linear_model import LinearRegression` `model = LinearRegression()` `model.fit(X_train,y_train)`
5	3- Check the Quality of the Model Assume : SL = 0.05 check the quality of the model is: 1. Ensure your test score > train score (Model must be perform best on UNKNOWN DATA !!!) 2. Ensure your test score >= (1 - SL)	`print(model.score(X_train,y_train))` `print(model.score(X_test,y_test))` `0.9494673013344644` ➔ train score `0.9816423482070255` ➔ test score

5	Challenge: Try to get the best model with target of minimum 99% accuracy. Data Randomization always give different results!!!	``` from sklearn.linear_model import LinearRegression from sklearn.model_selection import train_test_split for i in range(1,100): X_train,X_test,y_train,y_test = train_test_split(features,label,test_size=0.2,random_state=i) model = LinearRegression() model.fit(X_train,y_train) train_score = model.score(X_train,y_train) test_score = model.score(X_test,y_test) if test_score > train_score: print("Test S {}, Train Score {}, RandomSeed {}".format(test_score,train_score,i)) ``` Test S 0.9695039421049821, Train Score 0.9545249190394052, RandomSeed 3 Test S 0.9631182154839475, Train Score 0.9528197369259258, RandomSeed 8 Test S 0.9816423482070255, Train Score 0.9494673013344644, RandomSeed 10 Test S 0.9606215790278543, Train Score 0.9527636176933665, RandomSeed 14 Test S 0.9835849730044817, Train Score 0.9460054870434312, RandomSeed 26 Test S 0.9636425773684422, Train Score 0.9527636606684406, RandomSeed 27 **Test S 0.9944092048209744, Train Score 0.9400496694274888, RandomSeed 30** Test S 0.9778242092591887, Train Score 0.9486350116716654, RandomSeed 37 Test S 0.9724794487377619, Train Score 0.9473317052697812, RandomSeed 38 Test S 0.9928344802911049, Train Score 0.9492886917497556, RandomSeed 39 Test S 0.9802519469633169, Train Score 0.9491742100347064, RandomSeed 41 Test S 0.9789129767378081, Train Score 0.948821675263085, RandomSeed 46 Test S 0.98399193890564, Train Score 0.9486450781125914, RandomSeed 47 Test S 0.980277279178695, Train Score 0.9500780390200971, RandomSeed 48 Test S 0.9608624689052039, Train Score 0.9541375225175409, RandomSeed 51 Test S 0.9743646706957547, Train Score 0.952756273050018, RandomSeed 52 Test S 0.9804067424885895, Train Score 0.9504872715098402, RandomSeed 56 Test S 0.9719509793938971, Train Score 0.9473987125707488, RandomSeed 62 Test S 0.95820089851047, Train Score 0.9505483928196958, RandomSeed 63 Test S 0.9588832495320915, Train Score 0.9562672856609079, RandomSeed 67 Test S 0.9791787060652751, Train Score 0.937932068950384, RandomSeed 68 Test S 0.9694792167947474, Train Score 0.9504137960985714, RandomSeed 71 Test S 0.9562771755752736, Train Score 0.9562030951258303, RandomSeed 72 Test S 0.981214310330871, Train Score 0.9453900863447221, RandomSeed 73 Test S 0.9618591691900452, Train Score 0.9553251075019685, RandomSeed 74 ... Test S 0.9676701872390631, Train Score 0.9529778812782739, RandomSeed 90 Test S 0.9793995823406391, Train Score 0.9469346629378338, RandomSeed 92 Test S 0.9682219576297961, Train Score 0.9534166513146052, RandomSeed 93 Test S 0.9676991009836634, Train Score 0.9514417860805683, RandomSeed 94
	Final Model	``` X_train,X_test,y_train,y_test = train_test_split(features,label,test_size=0.2,**random_state=30**) finalModel = LinearRegression() finalModel.fit(X_train,y_train) print(finalModel.score(X_train,y_train)) print(finalModel.score(X_test,y_test)) 0.9400496694274888 ➔ train Score 0.9944092048209744 ➔ test Score ```
	4. If Satisfied, perform Deployment	``` import pickle pickle.dump(finalModel , open('modelSalaryPredictor.nair' , 'wb')) ```