**INDEX**

1. Part 1 - Data Preprocessing
2. Part 2 - Regression: Simple Linear Regression, Multiple Linear Regression, Polynomial Regression, SVR, Decision Tree Regression, Random Forest Regression
3. Part 3 - Classification: Logistic Regression, K-NN, SVM, Kernel SVM, Naive Bayes, Decision Tree Classification, Random Forest Classification
4. Part 4 - Clustering: K-Means, Hierarchical Clustering
5. Part 5 - Association Rule Learning: Apriori, Eclat
6. Part 6 - Reinforcement Learning: Upper Confidence Bound, Thompson Sampling
7. Part 7 - Natural Language Processing: Bag-of-words model and algorithms for NLP
8. Part 8 - Deep Learning: Artificial Neural Networks, Convolutional Neural Networks
9. Part 9 - Dimensionality Reduction: PCA, LDA, Kernel PCA
10. Part 10 - Model Selection & Boosting: k-fold Cross Validation, Parameter Tuning, Grid Search, XGBoost

**PART ONE: DATA PRE-PROCESSING:**

1. **Importing Libraries:** These are always used for all models. Scikit learn is best library for machine learning, for neural nets Tensor flow can be used.
2. numpy: Work with arrays.

***Import numpy as np***

1. matplotlib :Use plotting tools : pyplot a sub module mainly used for plots

***Import matplotlib.pyplot as plt***

1. Pandas: import data and create Matrix features and dependent variable vector

***Import pandas as pd***

1. **Importing data:**
2. Function **read\_csv** from Pandas imports data and creates data frame i.e a tavle having all values

***dataset = pd.read\_csv(‘file\_name’)***

1. Create matrix of feature and dependent variable vector.First columns are independent variable and last column is dependent variable. **-1** in python means last coloumn. ***Iloc*** *mns locate indices*

***X = dataset.iloc[: , :-1].values***

***Y = dataset.iloc[: , -1].values***

1. **Class** is a model of something we want to build for e.g. construction plan. **Object** is product of the class, i.e for eg house is the product of the plan. **Method** is a tool used on the object to perform some task. For e.g. Pandas is a class i.e. a plan to work with arrays; object here is arrays and iloc is a method or fn of that class to locate indice.
2. **Handling missing data:**
3. Removing that observation: Used when we have large observation or Replace missing data with average, median or most frequent one whichever is suitable.
4. Scikit learn is a data science library having many functions, we first use **simple Imputer** class from **impute** module to create a function/object **imputer** which takes in nan i.e. null from np and replaces by mean.

***from sklearn.impute import SimpleImputer***

***imputer = SimpleImputer(missing\_values = np.nan, strategy = 'mean')***

1. **Fit** method connects matrix of data to the class i.e. it looks at the missing salary and replaces it with the mean. **Transform** method converts the matrix to modified matrix.

***imputer.fit(X[:, 1:3])***

***X[:, 1:3] = imputer.transform(X[:, 1:3])***

1. **Encoding categorical data:**

It is difficult for ML model to interpret categorical data and output data relationship as it may not understand the data it might try to correlate the way in which categorical data is written and try to pattern it. Hence defining it in no help to encode it just as a category. Encoding it straight as 0, 1, 2 might make it learn that there is an order in 0, 1, 2 and predict next value as 3 hence we use one hot encoding.

1. **One hot encoding**: Involves creating binary vector for each country, if there are 3 classes, create 3 columns and add zeros and ones. It is used for categorical variables. However, for binary values, we can use zeros and ones.
2. Classes require parenthesis.
3. ColumnTransformer requires two argument, first argument **transformers** requires 3 things type of transformation, type in that type of transformation, indexes of column to be encoded. **Remainders** require

***from sklearn.compose import ColumnTransformer***

***from sklearn.preprocessing import OneHotEncoder***

***ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0] )] ,remainder= 'passthrough')***

***X = np.array(ct.fit\_transform(X))***

***from sklearn.preprocessing import LabelEncoder***

***de = LabelEncoder()***

***y = de.fit\_transform(y)***

1. **Splitting data set into training and testing:**

To be applied b4 feature scaling. Feature scaling means making all the data in same range as ML model will ignore the smaller data. Test data should not be used before model creation and if feature scaling is done before separating it uses test data as well.

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 1)

1. **Feature scaling:**

Used sometimes, because other times the coeff adjust themselves to normalize the values of that variable.

Feature scaling can be done by Standardization or Normalization.

Xstd= (x-xmean)/std\_deviatn xnormal = (x-xmin)/(xmax-xmin)

Lies bet (-3,3) lies bet(0,1)

Normalization is used for normal distribution. Standardization works always.

Class **Standardscaler** is used from scikit learn.preprocessor. **Standard scaler class requires 2D array as input.**

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train[:, 3:] = sc.fit\_transform(X\_train[:, 3:])

X\_test[:, 3:] = sc.transform(X\_test[:, 3:])

**PART 2: REGRESSION**

Simple regression has one feature and one variable to be predicted, Multi linear regression has more features, Polynmoial deals with non linear regression and also SVR, DTR, RTR deals with non linear

1. **Simple regression:** It aims to **predict continuous** data.

Y = b0 + b1x

**Minimum sum of squares of diff between the actual and predicted value is determined to find the best fitting line.**

from sklearn.**linear\_model** import **LinearRegression**

regressor = LinearRegression()

regressor.**fit**(x\_train, y\_train)

y\_pred = regressor.**predict**(x\_test)

**plt**.**scatter**(x\_train, y\_train, color = 'red')

**plt.plot**(x\_train, regressor.predict(x\_train), color = 'blue')

**plt.title**('Salary vs noo of years (for trained data)')

**plt.xlabel**('no of years')

**plt.ylabel**('salary')

**plt.show**()

plt.scatter(x\_test, y\_test, color = 'red')

plt.plot(x\_train, regressor.predict(x\_train), color = 'blue')

plt.title('Salary vs noo of years (for trained data)')

plt.xlabel('no of years')

plt.ylabel('salary')

plt.show()

print(regressor.predict([[12]]))

print(regressor.coef\_)

print(regressor.intercept\_)

1. **Multiple linear regression:** Similar to simple linear but with more no of variable.

Y = b0 + b1x1 + b2x2 + …

**Assumptions of Linear regression:**

1. Linearity
2. Homoscedasticity: Homogeniety of variance.
3. Multivariate normality
4. Independence of errors
5. Lack of multicollinearity: One independent variable predicts other
6. **Dummy variables:** Encoding categorical variables into numeric variables. These numeric variables are called dummy variables. Don’t include all dummy variables into regression equation as it results in multicollinearity as sum of all Dummy variables = 1 is a constraint. **Always exclude one dummy variable.**
7. **P value and statistical significance:** While testing a hypothesis, we get a probability of obtaining the results equal to what we got and that probability should be > Significance level
8. **Filtering data to create model:** All given data cannot be used because it can confuse the model and not all mathematical values have significance.
9. **Method to build the model i.e. filtering** selecting variables
10. All – in: Use all variables, when we have prior knowledge or you have to
11. Backward elimination (stepwise regression) :
12. Select significance level
13. Fill model with all variables
14. Consider the predictor with highest p value and remove that if it is greater that SL. (*Significance level is the confidence in hypothesis i.e. x and y are not related with a confidence level of SL 5%)*
15. Rebuild the model without this and again do step 3
16. After all such variables are removed, model is ready
17. Forward selection (stepwise regression) :
18. Select the SL
19. Fit all regression models with each variable and select one with lowest p value
20. Use this variable and add one more(try all)
21. Consider one with lowest p value and if p < SL go to previous step and add third variable; else model is created
22. Bidirectional elimination (stepwise regression) :
23. Select significance level to stay and enter
24. Perform next step of forward selection i.e add variables for p< Slenter
25. Perform steps of backward elimination i.e. see if any of the 2 variables can be eliminated.
26. Go to step 2 unless no variables can be added or removed and model is created.
27. Score comparison

**Backward elimination is fastest and commonly used. We will be using that.**

**Multi linear regression class automatically avoids the dummy variable trap.**

**We need not use above methods to eliminate insignificant features, class automatically selects that.**

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test)

np.set\_printoptions(precision=2)

print(np.concatenate((y\_pred.reshape**(**len(y\_pred),1**)**, y\_test.reshape(len(y\_test),1)),1))

1. **Polynomial Regression:**

Polynomial regression is a type of linear regression because coeff are in linear form. It considers x1, x1^2, x1^3, x1^4 etc. as different features.

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

poly\_reg = PolynomialFeatures(degree = 4)

X\_poly = poly\_reg.fit\_transform(X)

lin\_reg\_2 = LinearRegression()

lin\_reg\_2.fit(X\_poly, y)

**Higher resolution** offers better visualization in a way that it reduces the sampling time from 1sec to 0.1sec

1. **SVR:**

SVR has an epsilon insensitive tube, with radius measured vertically and not perpendicular to tube. Error for the points in the tube is not considered but, for the points outside the tube their error is their **distance** to the outer surface of tube. We **minimize** these, hence scaling is required. But, I think SVR gives importance to extreme points which might result in incorrect model. But will be useful to reduce the domination of local data

To learn about nonlinear SVR, we require

1. SVM Intuition
2. Kernel SVM intuition
3. The kernel trick
4. Types of Kernel function
5. Nonlinear Kernel SVR

There are different types of kernels, **Linear kernels** learn linear relationships and **non linear kernels** like RBF (Gaussian Radial basis) for non linear relations.

y = Y.reshape(len(Y),1)

from sklearn.preprocessing import StandardScaler

sc\_x = StandardScaler()

sc\_y = StandardScaler()

x = sc\_x.fit\_transform(X)

y = sc\_y.fit\_transform(y)

from sklearn.svm import SVR

regressor = SVR(kernel ='rbf')

regressor.fit(x,y)

sc\_y.inverse\_transform(regressor.predict(sc\_X.transform([[6.5]])))

1. **Decision Tree:** CART is an umbrella consisting of classification tree and regression tree

Decision tree intuition is splitting the data into groups and if the new data falls within a group output is the averages of output of that group Randam forest or regression trees donot require the data scaling as thee is no mathematical equation, it splits the data

**Predict function requires a 2D array input hence we use 2 square brackets**

from sklearn.tree import DecisionTreeRegressor

regressor = DecisionTreeRegressor()

regressor.fit(X, y)

regressor.predict([[6.5]])

X\_grid = np.arange(min(X), max(X), 0.01)

X\_grid = X\_grid.reshape((len(X\_grid), 1))

plt.scatter(X, y, color = 'red')

plt.plot(X\_grid, regressor.predict(X\_grid), color = 'blue')

plt.title('Truth or Bluff (Decision Tree Regression)')

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()

1. **Random forest Regression**

It is a version of ensemble learning, Ensemble learning is use multiple algo or same algo multiple times to optimize.

1. Pick k data points from training set
2. Build a decision tree based on these k points
3. Choose no of trees to be build and repeat steps 1 & 2
4. Use all the trees to predict a value for new data point and average those. About 500 trees

from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n\_estimators = 10, random\_state = 0)

regressor.fit(X, y)

**Evaluating Regression models using R2 Squared:**

R2 = 1 -

= Sum (yi – yregre)

= Sum(yi - yavg)

Here, closer the num to 0, better the regression line fits.

Intuition is how better the regression line fits as compared to the mean line.

On increasing the no of variables, R2 always increases evenif variable is not related. Hence, it is not a right parameter. So we used adjusted R2

Adj R2 = 1 – ( 1 – R2)\*

Adjusted R2 penalizes for adding a variable. p is the no of regressors and n is the sample size.

**Template that we have made for testing all the models assumes that there are no categorical, missing data and first columns are dependent and last is independent.**

from sklearn

.metrics import mean\_squared\_error

mean\_squared\_error(y\_test, y\_pred)

from sklearn.metrics import r2\_score

r2\_score(y\_test, y\_pred)

**PART 3: CLASSIFICATION**

1. **Logistic Regression**
2. **K-Nearest Neighbors (K-NN)**
3. **Support Vector Machine (SVM)**
4. **Kernel SVM**
5. **Naive Bayes**
6. **Decision Tree Classification**
7. **Random Forest Classification**
8. **Logistic Regression**

**Definition:** We use a sigmoid function to convert the values of regression model into probability values. This probability leads to a decision about the given x. It assumes Linear model like linear regression

Supervised classification algo, classification when a threshold for decision is used.

Using Linear rmodel and sigmoid fn to convert the output in probabilities i.e 0 to 1 and predict binary values.

Predict\_proba returns probability of prediction. Logistic regression is a **linear classifier**. Hence, the separation curve is a straight line.

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 0)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = **confusion**\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

1. **KNN Algo**

**Knn:** Find min no of neighbours to be considered and based on distances check these neighbours and find which group has more no assign the new value to that.  **Can be used for missing values for both categorical and continuos data.**

1. Choose no of neighbours k in total
2. Consider the nearest K neighbours based on Eucledian or Manhatten distancs.
3. Count data points in each category
4. Assign data point to category with most no of neigbours

from sklearn.neighbors import KNeighborsClassifier

nn = KNeighborsClassifier(n\_neighbors=7)

nn.fit(X\_train, Y\_train)

1. **SVM:**

Uses max margin hyperplane i.e the separation line must be at max distance from sum of distance from 2 points. Other classification algo look for the most likely or most similar case but SVM looks for the most confusing case e.g. in apple and oranges it looks for the apples that look like oranges. And thus makes it more robust. **Uses a linearly seperable data**

1. **K SVM:**

Mapping to a higher dimension makes non linear data as linearly separable.

Use kernel function, irregular boundaries i.e are combinations of gaussians or other functions are created.

Functions are Gaussian kernel, sigmoid kernel,, Polynomial kernel, Laplace kernel

1. **Naïve Bayes**

Uses bayes theorem, i.e P(A/B)\*P(B) = P(B/A)\*P(A)

Using this calculate probability for all categories and find which is max

1. **Decision tree**
2. **Random forest**

**PART 4: CLUSTERING: UNSUPERVISED**

Clustering is similar to classification butt here we don’t know what are we looking for. We try to look at some segment and thus unknowing things may also popup.

1. **K MEANS:**

1 Consider k no of clusters

2 Select random k centroids and place them anywhere

3 Assign points to nearest centroids using perpendicular line theorem.

4 Calculate geometric centroid of points,

5 Go to step 3 if dist of centroid from geom centroid is less than E.

Random initialization trap is when you select initial centroid and they don’t converge to actual results due to their initial position. Hence k means ++ is used. Not explained but K means ++ is used.

To find no of clusters: **WCSS i.e.** within clusters sum of squares is the parameter to find optimal no of clusters but as no of clusters increase, WCSS obv decreases but using an elbow method shows a point where to stop.

1. **HIERARCHIAL CLUSTERING:**

Two types: Agglomerative & Divisive

Aglomerative:

1. Make each data point as cluster
2. Consider closest cluster and make them single cluster
3. Take two nearest clusters and form 1 cluster
4. Distance between clusters can be farthest point dist, Nearest point dist, Centroid dist, Avg dist of all points depending on type of problem

All these clusters, step by step are stored in dendograms. It has total memory in the form of graphs where x axis has data points and y axis has Eucledian distance.

**PART 5: ASSOCIATION: UNSUPERVISED**

**APRIORI ALGORITHM**

It is used to find out that someone buys x also buys y

This algo has 3 parts support, confidence and lift

Support: Prob of doing a task, (people watching abc movie)/total no of people

Confidence: People watching B after a / people watching a

Lift: confidence / support.

1. Set min support and confidence
2. Consider all subsets in transactions having higher support
3. Take all the rules having confidence greater than min confidence
4. Consider rules in decreasing order of lift

**ECLAT LEARNING**

Support: Here it is probability for watching / purchasing a set of movies or fruits

1. Set a minimum support
2. Take all subsets having support higher than min support
3. Arrange these in decreasing order of support.

**DEEP LEARNING**

Method of mimicking human brain to improve learning. Inputs are either normalized or standardized.

Activation functions used are

1. Step
2. Sigmoid
3. Rectifier
4. Hyperbolic Tangent

Common configuration of NN is the rectifier function in hidden layers and sigmoid function in output

Popular cost function is **0.5\*(ypred – yactual)2**

Curse of Dimensionality: As no of feature increases the data required is exponentially more.

**Gradient Descent:** Determine the slope if it is negative, you are going downwards and positive is upwards

**Stochastic Gradient descent differs from gradient descent i.e batch gradient in a way that GD takes the whole table inputs and adjusts weight REINFORCEMENT LEARNING and again does the same for no of times. However, SGD uses single row for every time to adjust weight. It thus avoids local min trap.**

**Mini batch gradient lies between two.**

Back propagation helps you to know which part of NN is responsible for the particular part of error. And helps to adjust all the weights simultaneously.

<https://drive.google.com/drive/folders/1OFNnrHRZPZ3unWdErjLHod8Ibv2FfG1d>

<https://drive.google.com/drive/folders/19gdkL2xaEsvCkRHbLgSYkkDEGEcX5cp_>

**Central limit theorem:** If a large amount of sample is considered from a population then mean of those samples is normally distributed irrespective of population distribution.

**Normal/ Gaussian distribution:** It is the probability distribution of data in a bell shaped curve which is symmetric about the mean. Its characteristics include Symmetric, unimodal, Equal mean, mode and median; Asymptotic.

**Standard deviation:**  It tells about the spread of the distribution; It is the root of **variance** because Variance is in units of square. (var = sum(x-xmean)2 / n )

**Covariance:** It tells the relation between 2 variable, positive means when one increases other increases as well and the negative means it decreases when one increases zero means either one is constant or has no relation. But it is not intuitionable as magnitude of covariance doesn’t specify anything for ex comsider covariance of x vs x its large but they are equal. It depends on the scale of the data

Covariance = sum [(x-xmean)(y-ymean)] / n

**Corelation:** It signifies the relationship of x and y. It is not scale dependent. It can be -1 for negative line and +1 for positive line. And p value tells the amount of confidence in correlation.

Correlation = Covariance / [(Std devtn X)\*(Std devtn Y)]

**Better than both is R2**

**What is data science?**

Data science is a domain where large volumes of data are dealt with modern tools and techniques to derive meaningful insights from data to make decisions more reliable.

**Data scientist, Analyst and engineer**

Engineer usually deals with arranging the data, making sure they are appropriate and available for data scientist and engineers. Analyst works with interpreting the static data with normal statistics and visualization whereas data scientist works with dynamic data like ads, social media platform.

**Supervised vs unsupervised vs Reinforcement learning**

| **Supervised** | **Unsupervised** | **Reinforcement** |
| --- | --- | --- |
| Labelled data | Unlabeled data | Interacts with environments |
| Used for prediction | Used for analysis | Reward based |
| Known no of classes | Unknown no of classes | No predefined data or supervision |
| Regression and Classification | Clustering and Association | Trial and error method |
| Has training data and predicts something specific | Understands data and dsnt look for something specific |  |

**Decision tree regression:**

Uses information entropy i.e a measure of how scattered information is and using that creates seperations. And split is by using boundary condition for percentage of points compared to overall points. We find these groups and make a decision tree. For pure non linear data

**Confusion matrix:**

If model predicts occurrence of an event it is positive and if it predicts nonoccurrence it is negative.

**Accuracy, Error rate, precision, recall, F-score**

**Accuracy**: Detected correctly/ total; Misleading considering the example of SPAM

**Error rate**: Detected incorrectly/total

**Precision**: How many positives % from what model predicted positives in total are correct

**Recall/ Sensitivity**: Total correct positives model suggested to actually total positives.

**F-score**: There is a trade off between precsion and recall for e.g. in fishing problem where we used fishing stick to catch fish, precision is high and when we use net, recall is high. Hence,, F1 score i.e. harmonic mean of precision and recall is used.

**CLASSIFICATION**

**Logistic regression**:

Supervised classification algo, classification when a threshold for decision is used.

Using Linear rmodel and sigmoid fn to convert the output in probabilities i.e 0 to 1 and predict binary values.

**Knn:** Find min no of neighbours to be considered and based on distances check these neighbours and find which group has more no assign the new value to that.  **Can be used for missing values for both categorical and continuos data.**

**SVM**:

Uses max margin hyperplane i.e the separation line must be at max distance from sum of distance from 2 points. Other classification algo look for the most likely or most similar case but SVM looks for the most confusing case e.g. in apple and oranges it looks for the apples that look like oranges. And thus makes it more robust. **Uses a linearly seperable data**

**K SVM:**

Mapping to a higher dimension makes non linear data as linearly separable.

Use kernel function, irregular boundaries i.e are combinations of gaussians or other functions are created.

Functions are Gaussian kernel, sigmoid kernel,, Polynomial kernel, Laplace kernel

**How to decide which method to use?**

Use pair plots from seaburn library

**What is information entropy**

It tells the amount of scattering in the data and is found by Summation of –plogp

**Bias and underfiiting** refer to over simplification of model due to more asumptions or considering less no of parameters. Linear and logistic regression usually fall for bias which makes them learn faster.

**Variance and overfitting** refer to high complexity due to more variables or less assumptions. It also learns noise. Decision tree, KNN result in overfitting

**What is overfitting in the model**

When model only uses localized data, it is unable to work with global data because it considers lots of variables, Methods to avoid that

Use global data and keep the model simple using significance level

Use regularization techniques and cross validation

**Regularisation:** Penalising the features **19**

**Cross validation**

Using permutations of testing and training data from the given set is Cross validation

**Univariate, Bivariate and multivariate**

Univariate is data with one variable, Purpose of univariate analysis is to describe the data and define trends in it or by using mean, variance, and minimum, maximum. Bivariate analysis is used for relation between 2 variables. Multivariant is extension of bivariate

**Feature selection methods**

FILTER METHODS: Information gain, Chi square test, Corelation test

Filter methods usually consider each element and see their correlation with target attribute

WRAPPER METHODS: Forward elimination, Backward elimination, Bidirectional elimination

Wrapper methods use multiple variable to check the correlation between variables and target

**Outlier**

Outliers are extreme data points and can be removed if garbage but Can be corrected by using non linear models or models not affected such as random forest

If mean and variance of a data are constant over time, that data is **Stationary**

**KNN** can have low error due to high complexity i.e. variance by increasing the no of neighbours and

**SVM** by increasing no of violations

**Resampling** is done when cross validating to determine performance.

**Law of large nos:** As the sample size grows, the estimated mean, variance and SD approaches to population mean, variance, SD

**Cofounding variable:** Variable that affects the dependent and independent variable

Selection bias, undercover bias, Survivorship bias

**Standardisation**  is more robust for Gaussian distribution over **normalization**

**Feature scaling**  is required for algos depending on distances and not for graphical based algos neither for algo with coeff. E.g. Knn, k means, kernel svm. (Linear regression, logistic give better performance)

**How to choose an algo:**

1. Size of data: Less data- high bias algo like Linear Regre, Naïve bias, Linear SVM when data is large use high variance like knn, Rt, DT
2. Accuracy of output:
3. Training time: SVM, NN, DT require high time
4. Linearity: Kernel SVM, NN, RF handle it easily
5. Features: SVM can handle large features with less no of data, Dt requires large no of data

8,9, 14