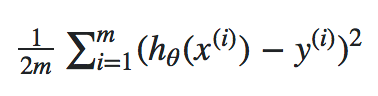
**Important Terms in ML**

1. Linear Regression – from sklearn.
2. Logistic Regression - performs binary classification

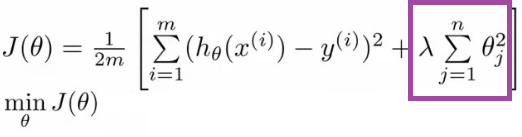
**The sigmoid/logistic function is given by the following equation: y = 1 / 1+ e-x**

1. Cost Function –

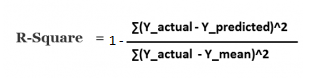


1. **Regularization** - Overfitting happens when model learns signal as well as noise in the training data and wouldn’t perform well on new data on which model wasn’t trained on. Now, there are few ways you can avoid overfitting your model on training data like **cross-validation sampling, reducing number of features, pruning, regularization** etc.

Regularization basically adds the penalty as model complexity increases. Regularization parameter (lambda) penalizes all the parameters except intercept so that model generalizes the data and won’t overfit.



1. R2 Score – Fitness of Good- Other metric to evaluate the performance of linear regression is R-square and most common metric to judge the performance of regression models. R² measures, “How much the change in output variable (y) is explained by the change in input variable(x).



R-squared is always between 0 and 1:

0 indicates that the model explains NIL variability in the response data around its mean.

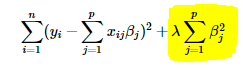
1 indicates that the model explains full variability in the response data around its mean.

In general, higher the R², more robust will be the model.

1. **LASSO Regression** - linear term (lambda) is added with the cost function also called as L1 Regularization. Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds “absolute value of magnitude” of coefficient as penalty term to the loss function.



1. **RIDGE Regression** – Squared term (lambda) is added with the cost function also called as L2 Regularization. Ridge regression adds “squared magnitude” of coefficient as penalty term to the loss function.



“Ridge regression” will use all predictors in final model whereas “Lasso regression” can be used for feature selection because coefficient values can be zero.

1. **Feature Scaling** – Rescale variables (e.g. standardizing or normalizing)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(df.drop('TARGET CLASS',axis=1))

scaled\_features = scaler.transform(df.drop('TARGET CLASS',axis=1))

df\_feat = pd.DataFrame(scaled\_features,columns=df.columns[:-1])

df\_feat.head()

1. **Cross Validation (K Fold Cross Validation)** – test data is splited into k subsets and each time k-1 subsets will be used for training the model and K- th subset is used to validate the model.
2. **Grid search** - <https://stackabuse.com/cross-validation-and-grid-search-for-model-selection-in-python/>

The first step you need to perform is to create a dictionary of all the parameters and their corresponding set of values that you want to test for best performance. The name of the dictionary items corresponds to the parameter name and the value corresponds to the list of values for the parameter.

grid\_param = {

'n\_estimators': [100, 300, 500, 800, 1000],

'criterion': ['gini', 'entropy'],

'bootstrap': [True, False]

}

The Grid Search algorithm basically tries all possible combinations of parameter values and returns the combination with the highest accuracy. For instance, in the above case the algorithm will check 20 combinations (5 x 2 x 2 = 20).

gd\_sr = GridSearchCV(estimator=classifier, param\_grid=grid\_param, scoring='accuracy', cv=5, n\_jobs=-1)

Once the parameter dictionary is created, the next step is to create an instance of the GridSearchCV class. You need to pass values for the estimator parameter, which basically is the algorithm that you want to execute. The param\_grid parameter takes the parameter dictionary that we just created as parameter, the scoring parameter takes the performance metrics, the cv parameter corresponds to number of folds, which is 5 in our case, and finally the n\_jobs parameter refers to the number of CPU's that you want to use for execution. A value of -1 for n\_jobs parameter means that use all available computing power.

Once the method completes execution, the next step is to check the parameters that return the highest accuracy. To do so, print the sr.best\_params\_ attribute of the GridSearchCVobject, as shown below:

best\_parameters = gd\_sr.best\_params\_ print(best\_parameters)

One of the major drawbacks of grid search is that when it comes to dimensionality, it suffers when the number of hyperparameters grows exponentially.

Random Search - Random search is a technique where random combinations of the hyperparameters are used to find the best solution for the built model. It tries random combinations of a range of values. To optimise with random search, the function is evaluated at some number of random configurations in the parameter space.

**Random Forest** **– How to choose next splitting node?**

It is a type of ensemble machine learning algorithm called Bootstrap Aggregation or bagging.

The forest that it builds is an ensemble of Decision Trees as we previously talk and most of the time it is trained with the “bagging” method. The basic concept behind the bagging method is that a combination of learning models increases the overall result.

**Naive – Bias Classifier** – Based on conditional probability. Best used with Categorical data.

Naive Bayes can be used in real-world applications such as:

Sentiment analysis and text classification

Recommendation systems like Netflix, Amazon

To mark an email as spam or not spam

Face recognition

**P( A | B) = P (B | A) \* P(A) / P(B)**

**SVC** - widely used in pattern recognition and classification problems — when your data has exactly two classes.

from sklearn.svm import SVC

SVM can be used in real-world applications such as:

detecting persons with common diseases such as diabetes hand-written character recognition

text categorization — news articles by topics stock market price prediction

The objective of the support vector machine algorithm is to find a hyperplane in N-dimensional space (N — the number of features) that distinctly classifies the data points.

**PCA with Mathematic Explaination** - Reduce data redundancy and dimensionality

<https://towardsdatascience.com/the-mathematics-behind-principal-component-analysis-fff2d7f4b643>

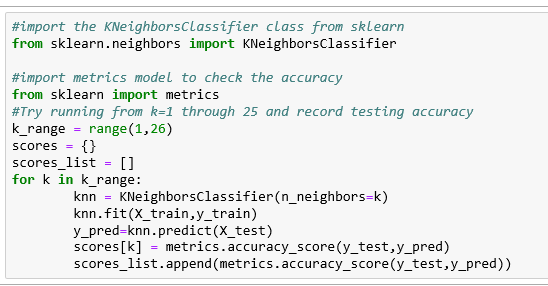
**Neural Network** -

**NLP** –

**Model Tuning** – Hyperparameters (user set manually)

**KNN** - <https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75>

from sklearn.neighbors import KNeighborsClassifier



**K-Means Clustering** - needs to know in advance how many clusters there will be in your data

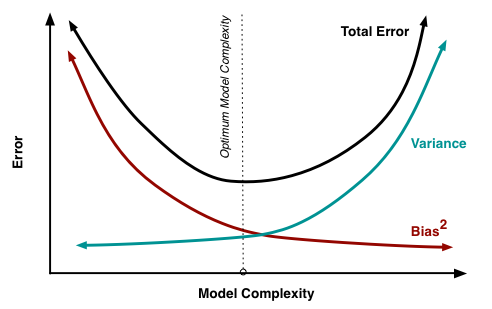
from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=4)

**Bias and Variance Trade-off** - There is a tradeoff between a model’s ability to minimize bias and variance.

If our model is too simple and has very few parameters, then it may have high bias and low variance. On the other hand, if our model has large number of parameters then it’s going to have high variance and low bias. So, we need to find the right/good balance without overfitting and underfitting the data.

**Err(x)=Bias\*\*2+Variance+Irreducible Error**



**Bias** - Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

**Variance** - Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn’t seen before. As a result, such models perform very well on training data but has high error rates on test data.

**Classification Report** - Report which includes Precision, Recall and F1-Score.

Accuracy

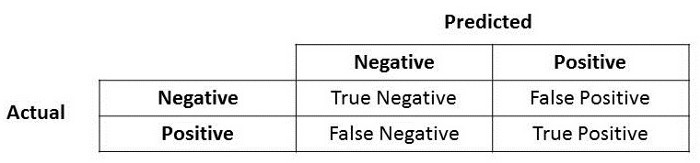
Precision –> Precision = TP/(TP + FP)

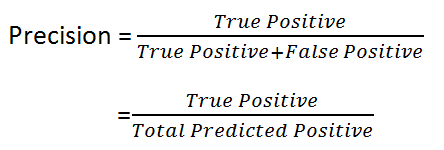
<https://joshlawman.com/metrics-classification-report-breakdown-precision-recall-f1/>

Recall -> Recall = TP/(TP+FN)

F1 Score => F1 = 2 x (precision x recall)/(precision + recall)

<https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9>





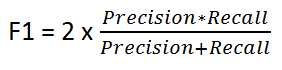
Precision is a good measure to determine, when the costs of False Positive is high. For instance, email spam detection. In email spam detection, a false positive means that an email that is non-spam (actual negative) has been identified as spam (predicted spam). The email user might lose important emails if the precision is not high for the spam detection model.

**Recall –** Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.

For instance, in fraud detection or sick patient detection. If a fraudulent transaction (Actual Positive) is predicted as non-fraudulent (Predicted Negative), the consequence can be very bad for the bank. Similarly, in sick patient detection. If a sick patient (Actual Positive) goes through the test and predicted as not sick (Predicted Negative).



**F1 Score:** F1 which is a function of Precision and Recall.



**We have previously seen that accuracy can be largely contributed by a large number of True Negatives which in most business circumstances, we do not focus on much whereas False Negative and False Positive usually has business costs (tangible & intangible) thus F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives).**

**Scaling the Data** –

from sklearn.preprocessing import StandardScaler

feature\_scaler = StandardScaler()

X\_train = feature\_scaler.fit\_transform(X\_train)

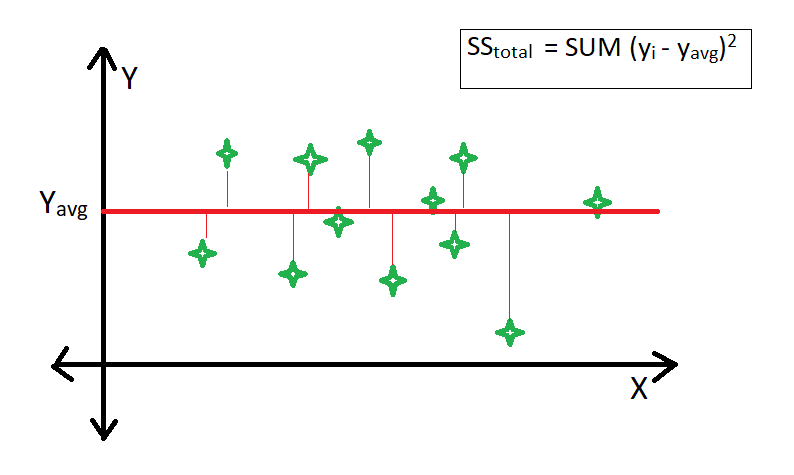
X\_test = feature\_scaler.transform(X\_test)

**R Square** – **R-squared** is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

R-squared or R2 explains the degree to which your input variables explain the variation of your output / predicted variable. So, if R-square is 0.8, it means 80% of the variation in the output variable is explained by the input variables. So, in simple terms, higher the R squared, the more variation is explained by your input variables and hence better is your model.

However, the problem with R-squared is that it will either stay the same or increase with addition of more variables, even if they do not have any relationship with the output variables. This is where “Adjusted R square” comes to help. Adjusted R-square penalizes you for adding variables which do not improve your existing model.

Hence, if you are building Linear regression on multiple variable, it is always suggested that you use Adjusted R-squared to judge goodness of model. In case you only have one input variable, R-square and Adjusted R squared would be exactly same.

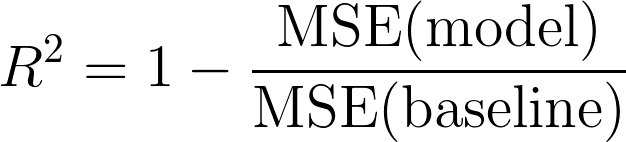


SStotal = SUM ( Yi – Yavg) \*\*2

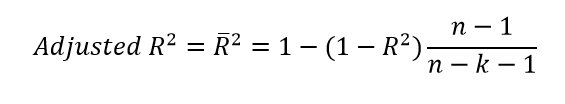
SSres = SUM (Yi – Ypred)\*\*2

https://media.geeksforgeeks.org/wp-content/uploads/20190415232942/CodeCogsEqn-10.gif

**In conclusion, R² is the ratio between how good our model is vs how good is the naive mean model.**



Adjusted R squared - The Adjusted R-squared value is similar to the R-squared value,but it accounts for the number of variables. This means that the Multiple R-squared will always increase.  
when a new variable is added to the prediction model, but if the variable is a non-significant one, the Adjusted R-squared value will decrease. Adjusted R² is a modified version of R² adjusted with the number of predictors. It penalizes for adding unnecessary features and allows a comparison of regression models with a different number of predictors. <https://medium.com/analytics-vidhya/measuring-the-goodness-of-fit-r%C2%B2-versus-adjusted-r%C2%B2-1e8ed0b5784a>

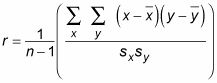


Here **k** is the number of explanatory variables in the model and **n** is the number of observations.

The value of adjusted R² is always less than that of R².

Coefficient of Correlation- <https://www.dummies.com/education/math/statistics/how-to-calculate-a-correlation/>

*Pearson correlation coefficient* is :



Correlation - <https://www.mathsisfun.com/data/correlation.html>

correlation formula onepass

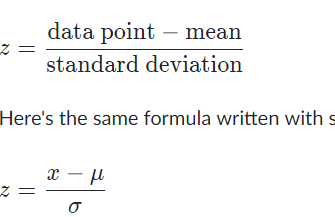
**Ordinary Least Square (OLS)** - Ordinary least squares regression (OLSR) is a generalized linear modeling technique. It is used for estimating all unknown parameters involved in a linear regression model, the goal of which is to minimize the sum of the squares of the difference of the observed variables and the explanatory variables.

<https://towardsdatascience.com/econometrics-behind-simple-linear-regression-ae5037de92c9>

Z score - Some of the methods of eliminating outliers are the Z-Score and the IQR Score methods.

A z-score measures exactly how many standard deviations above or below the mean a data point is.

Here's the formula for calculating a z-score:



Here are some important facts about z-scores:

A positive z-score says the data point is above average.

A negative z-score says the data point is below average.

A z-score close to 0 says the data point is close to average.

A data point can be considered unusual if its z-score is above 3 or below -3−3minus, 3.

**Confusion Matrix** – Type 1 Error and Type 2 Error

**Bootstrap Sampling** -

**Bagging** – OR **Bootstrap Aggregation** –

<https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/>

Bagging uses a simple approach that shows up in statistical analyses again and again — improve the estimate of one by combining the estimates of many. Bagging constructs n classification trees using bootstrap sampling of the training data and then combines their predictions to produce a final meta-prediction.

Sci-kit learn’s implementation of the bagging ensemble is BaggingClassifier.

from sklearn.ensemble import BaggingClassifier

from sklearn import tree

model = BaggingClassifier(tree.DecisionTreeClassifier(random\_state=1))

model.fit(x\_train, y\_train)

model.score(x\_test,y\_test)

**Boosting** - In short, it combines multiple weak or average predictors to a build strong predictor. These boosting algorithms always work well in data science competitions like Kaggle, AV Hackathon, CrowdAnalytix.

Boosted algorithms are used where we have plenty of data to make a prediction. And we seek exceptionally high predictive power. It is used for reducing bias and variance in supervised learning.

from sklearn.ensemble import GradientBoostingClassifier

model= GradientBoostingClassifier(learning\_rate=0.01,random\_state=1)

model.fit(x\_train, y\_train)

model.score(x\_test,y\_test)

XGBoost –

Gradient Decent Algo

Q1. What is **Feature Scaling** and what are the different ways to do it?

Ans. Most of the times, your dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Eucledian distance between two data points in their computations, this is a problem. To supress this effect, we need to bring all features to the same level of magnitudes. This can be acheived by scaling.

<https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e>

There are four common methods to perform Feature Scaling.

Standardisation: Standardisation replaces the values by their Z scores.

https://miro.medium.com/max/133/1*LysCPCvg0AzQenGoarL_hQ.png

This redistributes the features with their mean μ = 0 and standard deviation σ =1. **sklearn.preprocessing.scale** helps us implementing standardisation in python.

Mean Normalisation:

https://miro.medium.com/max/197/1*fyK4gMQrfJKV5pmbXSrNbg.png

This distribution will have values between -1 and 1with μ=0.

Standardisation and Mean Normalization can be used for algorithms that assumes zero centric data like Principal Component Analysis (PCA).

**Min-Max Scaling**: This scaling brings the value between 0 and 1.

https://miro.medium.com/max/188/1*19hq_t_NFQ6YVxMxsT0Cqg.png

**Unit Vector**: Scaling is done considering the whole feature vecture to be of unit length.

https://miro.medium.com/max/110/1*u2Up0eaer56dpmaElU3Zxw.png

When to Scale

Rule of thumb I follow here is any algorithm that computes distance or assumes normality, scale your features!!!

Eg- **k-nearest neighbors, Principal Component Analysis (PCA),** We can speed up**gradient descent** by scaling.

Tree based models are not distance based models and can handle varying ranges of features. Hence, Scaling is not required while modelling trees.

Algorithms like Linear Discriminant Analysis (LDA), Naive Bayes are by design equipped to handle this and gives weights to the features accordingly. Performing a feature scaling in these algorithms may not have much effect.

Q.2. **What are feature selection methods?**

Ans. <https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e>

Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model.

**Benefits of performing feature selection before modeling your data**

**Reduces Overfitting**

**Improves Accuracy**

**Reduces Training Time**

**Feature Selection Methods:**

Univariate Selection – Statistical tests can be used to select those features that have the strongest relationship with the output variable.

The scikit-learn library provides the [SelectKBest](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html?source=post_page---------------------------" \l "sklearn.feature_selection.SelectKBest) class that can be used with a suite of different statistical tests to select a specific number of features.

The example below uses the chi-squared (chi²) statistical test for non-negative features to select 10 of the best features from the Mobile Price Range Prediction Dataset.

from sklearn.feature\_selection import SelectKBest

#apply SelectKBest class to extract top 10 best features  
bestfeatures = SelectKBest(score\_func=chi2, k=10)  
fit = bestfeatures.fit(X,y)

Feature Importance- You can get the feature importance of each feature of your dataset by using the feature importance property of the model.

Feature importance is an inbuilt class that comes with Tree Based Classifiers, we will be using Extra Tree Classifier for extracting the top 10 features for the dataset.

from sklearn.ensemble import ExtraTreesClassifier  
import matplotlib.pyplot as plt  
model = ExtraTreesClassifier()  
model.fit(X,y)  
print(model.feature\_importances\_) #use inbuilt class feature\_importances of tree based classifiers  
#plot graph of feature importances for better visualization  
feat\_importances = pd.Series(model.feature\_importances\_, index=X.columns)  
feat\_importances.nlargest(10).plot(kind='barh')  
plt.show()

**Correlation Matrix with Heatmap** - Correlation states how the features are related to each other or the target variable.

corrmat = data.corr()  
top\_corr\_features = corrmat.index  
plt.figure(figsize=(20,20))  
#plot heat map  
g=sns.heatmap(data[top\_corr\_features].corr(),annot=True,cmap="RdYlGn")

**Label Encoding** - Sklearn provides a very efficient tool for encoding the levels of a categorical features into numeric values. LabelEncoder encode labels with value between 0 and n\_classes-1.

>> from sklearn.preprocessing import LabelEncoder

>> le=LabelEncoder()

# Iterating over all the common columns in train and test

>> for col in X\_test.columns.values:

# Encoding only categorical variables

if X\_test[col].dtypes=='object':

# Using whole data to form an exhaustive list of levels

data=X\_train[col].append(X\_test[col])

le.fit(data.values)

X\_train[col]=le.transform(X\_train[col])

X\_test[col]=le.transform(X\_test[col])

**One-Hot Encoding**- One-Hot Encoding transforms each categorical feature with n possible values into n binary features, with only one active.

<http://www.insightsbot.com/blog/McTKK/python-one-hot-encoding-with-scikit-learn>

>> from sklearn.preprocessing import OneHotEncoder

>> enc=OneHotEncoder(sparse=False)

1.    **What are assumptions of Linear Regression? What are the most common estimators for linear regression?**

2.    What is the formula for logistic regression? How it is used for binary classification?

3.    How the decision tree decides on its split?

4.    What advantages does a Decision Tree model have?

5.    What is the difference between a Random Forest and versus Boosting Tree Algorithms?

6.    Given a dataset with features X and Labels Y, what assumptions are made when using Naive Bayes Theorem?

7.    What is Overfitting and what causes it? What ways can you use to avoid Overfitting?

8.    Describe the difference between Accuracy, Precision and Recall?

9.    What metrics can be used to evaluate a Regression task?

10. Describe how the Support Vector Machine (SVM) algorithm works?

Q. How the decision tree decides on its split?

Ans. <https://clearpredictions.com/Home/DecisionTree>

Gini Index

Gini index says, if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.

It works with categorical target variable “Success” or “Failure”.

It performs only Binary splits

Higher the value of Gini higher the homogeneity.

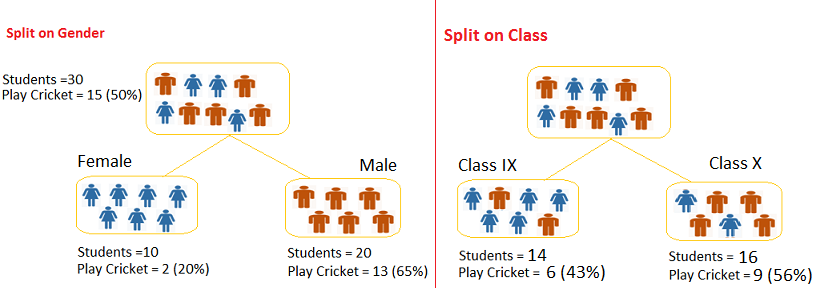
CART (Classification and Regression Tree) uses Gini method to create binary splits.

**Steps to Calculate Gini for a split**

Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure (p^2+q^2).

Calculate Gini for split using weighted Gini score of each node of that split

**Example: –**Referring to example used above, where we want to segregate the students based on target variable ( playing cricket or not ). In the snapshot below, we split the population using two input variables Gender and Class. Now, I want to identify which split is producing more homogeneous sub-nodes using Gini index.



Calculate, Gini for sub-node Female = (0.2)\*(0.2)+(0.8)\*(0.8)=0.68

Gini for sub-node Male = (0.65)\*(0.65)+(0.35)\*(0.35)=0.55

Calculate weighted Gini for Split Gender = (10/30)\*0.68+(20/30)\*0.55 = 0.59

Similar for Split on Class:

Gini for sub-node Class IX = (0.43)\*(0.43)+(0.57)\*(0.57)=0.51

Gini for sub-node Class X = (0.56)\*(0.56)+(0.44)\*(0.44)=0.51

Calculate weighted Gini for Split Class = (14/30)\*0.51+(16/30)\*0.51 = 0.51

Above, you can see that Gini score for *Split on Gender* is higher than *Split on Class,* hence, the node split will take place on Gender.

2. Chi-Square - <https://www.mathsisfun.com/data/chi-square-test.html>

It is an algorithm to find out the statistical significance between the differences between sub-nodes and parent node. We measure it by sum of squares of standardized differences between observed and expected frequencies of target variable.

It works with categorical target variable “Success” or “Failure”.

It can perform two or more splits.

Higher the value of Chi-Square higher the statistical significance of differences between sub-node and Parent node.

Chi-Square of each node is calculated using formula,

Chi-square = ((Actual – Expected)^2 / Expected)^1/2

It generates tree called CHAID (Chi-square Automatic Interaction Detector)

**Steps to Calculate Chi-square for a split:**

Calculate Chi-square for individual node by calculating the deviation for Success and Failure both

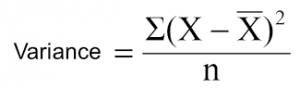
Calculated Chi-square of Split using Sum of all Chi-square of success and Failure of each node of the split

3. Information Gain:

Look at the image below and think which node can be described easily. I am sure, your answer is C because it requires less information as all values are similar. On the other hand, B requires more information to describe it and A requires the maximum information. In other words, we can say that C is a Pure node, B is less Impure and A is more impure.

4. Reduction in Variance

Till now, we have discussed the algorithms for categorical target variable. Reduction in variance is an algorithm used for continuous target variables (regression problems). This algorithm uses the standard formula of variance to choose the best split. The split with lower variance is selected as the criteria to split the population:



Above X-bar is mean of the values, X is actual and n is number of values.

**Steps to calculate Variance:**

Calculate variance for each node.

Calculate variance for each split as weighted average of each node variance.

Q. Pricing Analytics

**Machine Learning Algorithms and Python Packages**

* Linear Regression
* Model Validation
* Logistic Regression
* Train – Test Split
* SVM - from sklearn.svm import SVC  
   clf = SVC(kernel='linear')  
   clf.fit(X, y)
* PCA
* Decision Trees ----
* from sklearn.tree import DecisionTreeClassifier
* KNN
* K Means Clustering
* Random Forest
* NLP
* Model Validation
* Confusion Matrix
* Classification Report

**Python**

1. List comprehension
2. Lambda expression
3. Fibonaci series
4. Generator
5. Class
6. Vector
7. Array
8. Pandas library
9. Functions
10. Converting categorial variables to numeric
11. Dataframe manipulation using Pandas

100+ Python challenging programming exercises

Q. what are Pipelines in python?

**Pipelines** are a simple way to keep your data preprocessing and modeling code organized. Specifically, a pipeline bundles preprocessing and modeling steps so you can use the whole bundle as if it were a single step.

Factorial Program ?

|  |
| --- |
| def fact(x): |
| if x == 0: |
| return 1 |
| return x \* fact(x - 1) |
|  |
| x=int(raw\_input()) |
| print fact(x) |

**Statistics**

1. **Standard Deviation and Vairance -** [**https://www.mathsisfun.com/data/standard-deviation.html**](https://www.mathsisfun.com/data/standard-deviation.html)
2. **Standard deviation for a sample is calculated by dividing n-1 instead of n, why?**

[**https://www.graphpad.com/support/faqid/1383/**](https://www.graphpad.com/support/faqid/1383/)

[**https://towardsdatascience.com/why-sample-variance-is-divided-by-n-1-89821b83ef6d**](https://towardsdatascience.com/why-sample-variance-is-divided-by-n-1-89821b83ef6d)

1. **Binomial Distribution-** [**https://www.mathsisfun.com/data/binomial-distribution.html**](https://www.mathsisfun.com/data/binomial-distribution.html)
2. **Mean, Median, Mode**
3. **Kurtosis**
4. **Skewness**
5. **Central limit theorem** - The central limit theorem states that the distribution of sample means approximates a normal distribution as the sample size gets larger (assuming that all samples are identical in size), regardless of population distribution shape
6. Normal Distribution
7. Binomial Distribution
8. Poisson Distribution
9. Hypothesis Testing
10. T- Test
11. Z- Test
12. Significance level
13. P value –
14. Confidence interval

**Mathematics**

1. Probability, conditional probability
2. Matrix and Vector Operations
3. Eigen Vector
4. Transpose, Determinant,
5. Logarithms -<https://www.mathsisfun.com/algebra/logarithms.html>
6. Calculus