Understand the data

If regression, check correlation between y and continuous x values

If classification, check if data balanced

Check numeric, categorical and ordinal values

Check the range and distribution of the values

If any time component present

If numeric, but might be representing categorical already. Need to label encode those.

Check if any missing values

Understand the context of the data

Outliers (univariate: z score +-3 for normal distribution; else Q1-1.5IQR <-> Q3+1.5IQR. bivariate: scatter plot)

Multi correlation presence. Not only pearson correlation. Check also the distribution of each variable and if not Gausssian. Go for spearsman R correlation coefficient using dendogram.

Spearsmanr => from stats import spearsmanr. Output gives coef and p. If p > 0.05 then uncorrelated.

Go for clustering to get more insights of the data

Check for auto correlation

Understand each variable and also decide if binning or any other transformation is needed

Create more variables if needed

Linear Regression: Assumptions and Solutions:

<https://pythonfordatascience.org/linear-regression-python/>

* No Multicollinearity:
  + Multicollinearity is a phenomenon in which one predictor variable in a multiple regression model can be linearly predicted from the others with a substantial degree of accuracy.
  + Df.corr gives the correlation scores
  + Solution:
    - “variance inflation factor” (VIF) can be used to eliminate collinearity among the features. **Remove the one with correlation index > 6 or 7**
    - PCA can address the same as well however with it especially reduces the dimensionality and with little bit of information loss

Ref:

https://www.kaggle.com/ffisegydd/sklearn-multicollinearity-class

<https://www.kaggle.com/robertoruiz/dealing-with-multicollinearity>

* Linear Relationship Between Target & Features:
  + For simple linear regression: sse and sst should not be high enough. r^2 and adj\_r^2n should be well close to 1
  + Take each IV and plot against V to see if linearity exists
  + With multiple features: Create residual plot. Check normality of residuals.
  + If not, Solution:
    - Transform the features (log(x or y), 1/y, sqrt(y))
    - Apply different algorithm
* No Autocorrelation:
  + The analysis of autocorrelation is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal obscured by noise. When the residuals are autocorrelated, it means that the current value is dependent of the previous (historic) values.
  + Look for Durbin – Watson (DW) statistic (from statsmodels package). It must lie between 0 and 4. If DW = 2, implies no autocorrelation, 0 < DW < 2 implies positive autocorrelation while 2 < DW < 4 indicates negative autocorrelation
  + Solution:
    - Forget Linear Regression and consider time series
* No Outliers:
  + Solution:
    - Check if outliers are valid cases. Accordingly take action. If substantial number of outliers then treat them as a different dataset
    - Remove or impute the outliers, if invalid
* No High-Leverage Points:
  + Leverage is a measure of how far away the independent variable values of an observation are from those of the other observations. Difference between outliers and high-leverage points - <https://onlinecourses.science.psu.edu/stat501/node/337/>
  + Solution:
    - Check if those are valid cases and act accordingly.
    - Remove or impute those if invalid
    - Consider more robust algorithm

Then apply multiple linear regression to get the residuals using predicted y values.

* Homoscedasticity of Error Terms (Constant Variance of errors):
  + Homescedasticity means the errors exhibit constant variance. Heteroscedasticity, on the other hand, is what happens when errors show some sort of growth
  + Plot between each of X and residuals. If it follows funnel like shape then homoscedasticity is there.
  + The points should lie on the line
  + Or do levene’s test
  + Solution:
    - Consider log transformation, square or square root of the target values
* Normal Distribution of error terms:
  + If the error terms are non- normally distributed, confidence intervals may become too wide or narrow. We can look at QQ plot (shown below) to check the distribution
  + Solution:
    - If the errors are not normally distributed, non – linear transformation of the variables (response or predictors) can bring improvement in the model

Probability plot of residuals or Kolmogorov-Smirnov test for normality check

The number of observations must be greater than number of Xs

The mean of residuals is zero

<https://dziganto.github.io/data%20science/linear%20regression/machine%20learning/python/Linear-Regression-101-Assumptions-and-Evaluation/>

<https://www.analyticsvidhya.com/blog/2015/10/regression-python-beginners/>

OLS and Gradient Descent based linear models:

<https://stackoverflow.com/questions/34469237/linear-regression-and-gradient-descent-in-scikit-learn-pandas>

<https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model>

Regularization:

For any machine learning problem, essentially, you can break your data points into two components — **pattern + stochastic noise**.

For instance, if you were to model the price of an apartment, you know that the price depends on the area of the apartment, no. of bedrooms, etc. So those factors contribute to the pattern — more bedrooms would typically lead to higher prices. However, all apartments with the same area and no. of bedrooms do not have the exact same price. The variation in price is the noise.

The model complexity depends on the magnitude of the coefficients. The values of the coefficients or weights decide how well the model is fit on a set of data.

L2 -> Adds squared magnitude of the coefficient as penalty term in the loss function

L1 -> Adds the absolute value of the coefficient as penalty term in the loss function

<https://www.quora.com/What-is-regularization-in-machine-learning>

<https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>

Ridge

Good for highly multi correlated data

Uses L2 regularization

Does not shrink the coefficients to zero, hence no feature selection. Because of square of the weight, the partial differentiation of it (a part of modified cost function) results in division of it by a factor. The number can get reduced to very small but never zero.

Lasso

Good for highly multi correlated data

Uses L1 regularization

Shrinks the coefficients to zero, hence does help with feature selection (with non-zero coefficients)

Both Ridge and Lasso are gradient descent based algorithms.

In Lasso **max\_iter** is more importance than in Ridge. Max\_iter = number of iterations model has to run if it does not converge.

Max\_iter: This is the maximum number of iterations for which we want the model to run if it doesn’t converge before.

<https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b>

<https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/>

<https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/>

No assumptions for ridge and lasso:

<https://stats.stackexchange.com/questions/169664/what-are-the-assumptions-of-ridge-regression-and-how-to-test-them>

Assumptions on SGD? Gradient Descent is independent of assumptions. It is more about the linear model.

<https://stats.stackexchange.com/questions/266406/assumptions-of-linear-regression-and-gradient-descent>

**List of Linear Models**

Simple Linear Regression

Logistic Regression

Lasso Regression (L1)

Ridge Regression (L2)

Elastic Net

Naïve Bayes

Most important is to check the loss function for the requirement at hand.

Basic Metrics To Assess Linear Model:

* Sum of Squared Errors (SSE)
  + SSE is measure of how far off our model’s predictions are from the observed values. If you think about what squaring does to large numbers, you’ll realize that we’re really penalizing large errors. It’s like saying, it’s okay to miss on the close points but don’t allow large deviations between the model and the most distant points
* Total Sum of Squares (SST)
  + SST is a measure of the variance in the target variable. It has nothing do with the predicted values.
* R^2
  + R^2 measures how much variance is captured by the model. The range for Ordinary Least Squares is [0,1]. It is possible to get negative values for R^2 but that would require a fitting procedure other than OLS or non-linear data. Always know your assumptions!
* Adjusted R^2
  + Adjusted R^2 is the same as standard R^2 except that it penalizes models when additional features are added.

Why R^2 is a Poor Metric:

* R^2 will only go up as additional features are added, even if those features provide the model no new predictive power. However, adjusted R^2 levels out because of the penalty involved

Ref:

<https://dziganto.github.io/data%20science/linear%20regression/machine%20learning/python/Linear-Regression-101-Metrics/>

How to know if the data is linearly separable (Basically for classification problem):

* Easy method is to do data visualization if the dimension is as low as 2. However for data with higher dimensions following tests can be done.
  + So one thing that is feasible to do is if the accuracy of non-linear SVM classifiers is much better than the linear SVM classifiers, then we can infer that the data set is not linearly separable. Otherwise, its the other way around.
  + Use a perceptron to determine the same

Ref: <http://www.tarekatwan.com/index.php/2017/12/methods-for-testing-linear-separability-in-python/>

<https://discuss.analyticsvidhya.com/t/machine-learning-algorithms-linear-vs-non-linear/14990/2>

Naïve Bayes model:

Assumption: There should not be any correlation between two features in the input

Note: It is useful for huge dataset

Difference between parametric and non-parametric models:

Parametric: Assumes distribution of the data. Solely dependent on the parameters values to make inference.

Non-parametric – Does not hold any assumption. Approximates the states of all the data so for inference it is dependent on the state of the new data.

<https://www.quora.com/What-is-the-difference-between-a-parametric-model-and-a-non-parametric-model>

Decision Tree

<https://www.python-course.eu/Decision_Trees.php>

<https://www.python-course.eu/Regression_Trees.php>

<https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb>

**Max\_Depth**: Length of the longest path from a root to a leaf in a tree

Size: The number of nodes in the tree. If binary split then (2\*\*d+1) – 1, where d = depth

**Max\_features** = p/3 for regression and sqrt of p for classification. The features are always randomly permuted at each split. Therefore, the best found split may vary, even with the same training data and max\_features=n\_features, if the improvement of the criterion is identical for several splits enumerated during the search of the best split

If int, then consider max\_features features at each split.

If float, then max\_features is a fraction and int(max\_features \* n\_features) features are considered at each split.

If “sqrt”, then max\_features=sqrt(n\_features)

**Min\_samples\_Split** = min nbr of samples for a split to be considered

**Min\_samples\_leaf** = min nbr of samples each leaf node should contain

Hyperparameters in bold need to be optimized.

The ones in blue are to be tuned to decide the size of the tree

Graphviz used for visualization

**How split happens in the tree:**

The split of a node increases homogeneity of the resulting sub nodes. By default the tree will split until each node has one element. However certain algorithms are there for split and threshold can be defined as well.

For all the features and their values splits are done to check information gain (difference between entropies before and after split), the split with the best IG is considered. If gini is selected, same is the split logic.

Classification: Gini, Information Gain (difference in entropy)

Regression: Change in Variance

Gini: The Gini Index is calculated by subtracting the sum of the squared probabilities of each class from one. Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure (p²+q²). It is the probability of a row being misclassified if we randomly pick a label from a branch. If the sample is completely homogeneous, then the gini is zero and if the sample is an equally divided (50% – 50%), it has gini score of 0.5

<https://www.quora.com/What-is-the-interpretation-and-intuitive-explanation-of-Gini-impurity-in-decision-trees>

Information Gain(Entropy): Information Gain is the Entropy of the parent node minus the entropy of the child nodes. Entropy is sum of (class prob \* log of prob with base 2). If the sample is completely homogeneous, then the entropy is zero and if the sample is an equally divided (50% – 50%), it has entropy of one.

<http://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/>

<https://math.stackexchange.com/questions/331103/intuitive-explanation-of-entropy>

Gini or entropy does not matter much. However, if we want we can do a testing. In fact, entropy adds more computation because of log calculation of values.

we can use the variance of a feature regarding the target feature as splitting criteria in regression problem instead of the information gain. We then use the feature with the lowest weighted variance as splitting feature

**Pruning:**

Pruning is basically stopping the split in Decision Trees. In pre-puning the split is stopped early and post-puning the very specific branches are deleted. Technique of stopping a split or deleting it, to solve overfitting problem. In pre-prun **min\_impurity\_decrease** is used as a threshold for a split. Post prun is more effective than pre-prun. Sometimes a split might result in –ve gain but further split results in more +ve. In that case we might want to keep both the splits and that is possible using post pruning only. **Max\_leaf\_nodes, min\_sample\_leaf, max\_depth** are parameters of pruning. Moreover granular pruning control can be implemented by converting decision nodes to leaf nodes with **TREE\_LEAF**.

<https://stackoverflow.com/questions/49428469/pruning-decision-trees>

<https://statinfer.com/204-3-10-pruning-a-decision-tree-in-python/>

<https://www.saedsayad.com/decision_tree_overfitting.htm>

Drawbacks:

ID3 or CART are prone to overfitting If allowed to reach max depth. Although pruning can be done with controlling parameters, this model is unstable as a small change in the training data will result in a very difference model.

Random Forest:

The forest chooses the classification having the most votes (over all the trees in the forest) and in case of regression, it takes the average of outputs by different trees. Samples of the dataset are taken randomly with replacement.

<https://www.analyticsvidhya.com/blog/tag/jeremy-howard/>

Hyperparameters:

Pruning should not be done in RF. It is for decision tree. To reduce overfitting in RF, we should rather focus on reducing depth.

Optimization:

N\_estimators – Till 300. Higher the better accuracy. But, adds more computation.

Max\_depth – Till 32, 50 (with more depth, more overfitting)

Max\_features – sqrt of total nbr of features

Min\_sample\_leaf – 3, 5, 10, 25

Min\_sample\_split

Bootstrap = sampling with replacement. In every sample selects nearly 2/3 of n randomly (<https://stats.stackexchange.com/questions/347818/number-of-samples-per-tree-in-a-random-forest>)

Interpretation:

Feature importance

Hierarchy cluster using ranking correlation coefficient (spearsman r) to find redundant features

Partial dependence using plot\_pdp

Treeinterpreter for finding prediction path a row

<https://www.analyticsvidhya.com/blog/2018/10/interpret-random-forest-model-machine-learning-programmers/>

SVM:

Support vector machine is a linear model and it always looks for a hyperplane to separate one class from another. Support Vector Machine is responsible for finding the decision boundary to separate different classes and maximize the margin. Margin is the distance between the decision boundry and the closest data point.

<https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496>

<https://stackoverflow.com/questions/33778297/support-vector-machine-kernel-types>

<https://chrisalbon.com/machine_learning/support_vector_machines/svc_parameters_using_rbf_kernel/>

<https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>

How to choose hyperplan: From all the imaginary hyperplans in the feature space, the one that correctly classifies and has maximum distance from the nearest data point is considered.

If RGF, then gamma (how correctly classification is done) and C (how tolerant to misclassification) need to be optimized.

Gamma = 0.01

C = 1

Linear -> poly -> rbf (radial basis function)

Kernels:

Liner: When linearly separable

Else:

Gaussian

RBF

Anova

<https://stackoverflow.com/questions/33778297/support-vector-machine-kernel-types>

Classification Loss Functions:

**Precision**: TP/TP+FP (If score is 1 then that means all the negatives predicted correctly although some positives not)

Cannot afford any false alarm like irrelevant result in search engine

**Recall**: TP/TP+FN (If score is 1 then that means all the positives are predicted correctly. Depending upon the problem, higher the score better is the prediction powder)

Accuracy: Gives equal weightage to both –ve and +ve. It just cares about how many got predicted correctly from the entire population. For an imbalanced dataset this will be misleading score to rely on.

Recall/Sensitivity: True positive rate (TP/(TP+FN))

Like predicting cancer because cannot afford false negative case

**Specificity**: True Negative rate (TN/(TN+FP))

1 – Specificity => False positive rate

**F1 score**: This is a weighted mean of the both precision and recall. Hence we need a balance between precision and recall and given dataset is not balanced (large number of actual negatives) then going for F1 score makes more sense than accuracy.

2 \* (recall \* precision) / (recall + precision) = 2TP/(2TP + FP + FN)

Harmonic mean. Much appropriate when data is imbalanced.

Sensitive to the threshold value.

If unbalanced data, then f1 score is appropriate to evaluate a model.

**MCC**:

Mathews Correlation Coefficient

It truly considers the ratio of confusion matrix values to create a balanced measure even if the dataset is highly imbalanced.

((TP\*TN) – (FP\*FN))/sqrt((TP+FP) \* (TP+FN) \* (TN+FP) \* (TN+FN))

Values ranges from -1 to +1. With +1 denoting the best model and -1 the worst.

F1 score does care about which majority class is defined as positive, but MCC does not.

In highly unbalanced dataset,

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5721660/>

<https://en.wikipedia.org/wiki/Matthews_correlation_coefficient>

<https://lettier.github.io/posts/2016-08-05-matthews-correlation-coefficient.html>

**AUC ROC**: Recall vs (1-specificity) for various thresholds (between 0 and 1)

ROC: Receiver Operating Characteristic = Probablity curve = it summarizes all of the confusion matrices for each threshold values

<https://www.youtube.com/watch?v=xugjARegisk>

If the curve is closer to the sensitivity (y axis), then better is the result. Of course, higher AUC magnitude matters.

Not sensitive the threshold value. It just cares about the rank of prediction values, not the absolute values.

<https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>

<https://stackoverflow.com/questions/44172162/f1-score-vs-roc-auc>

**Log loss**:

Considers the difference between each prediction and its corresponding true value (based on probabilistic difference). Higher the value of log loss, worse the model is. Baseline model has log loss of 0.69. Hence a model with higher than 0.6 is not a good model at all.

Not suitable for imbalanced data.

If balanced data, then log loss is the best approach to evaluate strength of a model.

<https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a>

**Comparing two models:**

If balanced, then log loss

If unbalanced, then f1 score

* high AUC ROC vs low f1 or other "point" metric, means that **your classifier currently does a bad job, however you can find the threshold for which its score is actually pretty decent**
* low AUC ROC and low f1 or other "point" metric, means that **your classifier currently does a bad job, and even fitting a threshold will not change it**
* high AUC ROC and high f1 or other "point" metric, means that **your classifier currently does a decent job, and for many other values of threshold it would do the same**
* low AUC ROC vs high f1 or other "point" metric, means that **your classifier currently does a decent job, however for many other values of threshold - it is pretty bad**

**Multi class problem:**

Recall, precision, f1 score and log loss…. But no AUC ROC

Micro average – give equal weight at element level. Larger set of values dominate.

Macro average – calculate at the class level giving equal weights to at the element level

If unbalanced then macro average

All this passed as arguments in f1-score calculation

If balanced then log loss (also called as cross entropy loss)

For AUC ROC, need to binarize the outputs (One Vs All)

<https://medium.com/usf-msds/choosing-the-right-metric-for-evaluating-machine-learning-models-part-2-86d5649a5428>

**Difference Between R square and adjusted R square**

The only drawback of R2is that if new predictors (X) are added to our model, R2 only increases or remains constant but it never decreases. We can not judge that by increasing complexity of our model, are we making it more accurate?

That is why, we use “Adjusted R-Square”.

**Penalize more for wrong classification for certain classes**

**Update cross entropy loss function to penalize more for certain classes.**

<https://github.com/keras-team/keras/issues/2115>

<https://github.com/keras-team/keras/issues/3068>

<https://datascience.stackexchange.com/questions/32071/how-to-set-class-weight-for-imbalanced-classes-in-kerasclassifier-while-it-is-us>

**Handling imbalanced data and how evaluation is impacted**

Check if time related values are there. If yes, then simply class\_weights

If not, then ensemble cross validation/over sampling/class\_weights/

**Oversampling**

SMOTE in Python (Creates synthetic near duplicate entries using K means clustering)

from imblearn.over\_sampling import SMOTE

Should be done before train and val split

For Undersampling

**Nearmiss**

from imblearn.over\_sampling import RandomOverSampler

This helps in just creating duplicates without any alteration

<https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets>

**Learning Rate**

<https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1>

LDA for topic modelling

Every document is a collection of topics

Every topic collection of words

It arranges the distribution of topics in document and words in topics

Document matrix (Doc vs word) converted to doc vs topic and topic vs word matrices

LDA updates the parameters of the matrices using probability based sampling techniques

Iterates through each word and document and adjusts the probability of each topic

Parameters:

Alpha, beta – topic density and word density

Number of topics

Number of topic terms

Number of iterations

<https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/>

PCA  
<https://towardsdatascience.com/a-step-by-step-explanation-of-principal-component-analysis-b836fb9c97e2>

It helps in dimension reduction.

It helps in faster training.

It helps in capturing interactive effects in the data.

<https://sebastianraschka.com/Articles/2014_pca_step_by_step.html>

<https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/multivariate/how-to/principal-components/interpret-the-results/key-results/>

Definition:

Projecting feature space to a smaller space that represents the data well. For this transformation, eigenvalues and vectors of the scatter matrix of all the features are used. It only considers the input space and ignores the labelled classes.

Steps:

Compute mean vector of each dimension and then create scatter matrix (matrix of samples-mean vector). Eigenvector and eigenvalue (explains length and direction of the vector as well) for each dimension in the scatter matrix. Sort the eigenvalues and consider only the higher ones.

Steps to select number of components (either of these):

1. cumulative explained variance vs nbr of components
2. eigenvalues greater than 1 (eigenvector v of a square matrix A is non-zero vector such that if the v is multiplied by the A, then only the result is just multiplication of v and a scalar. That scalar is eigenvalue)
3. nbr of clusters vs eigenvalues. The point at which the line starts becoming straight line

PCA and MDA (multiple discriminant analysis)

In other words, via PCA, we are projecting the entire set of data (without class labels) onto a different subspace, and in MDA, we are trying to determine a suitable subspace to distinguish between patterns that belong to different classes. Or, roughly speaking in PCA we are trying to find the axes with maximum variances where the data is most spread (within a class, since PCA treats the whole data set as one class), and in MDA we are additionally maximizing the spread between classes.

Model Deployment

1. To handle preprocessing at flask API level: Create one preprocessing pickle file like x.columns format and another pickle file for the model itself. Handle the missing values and then call the model
2. Another way is to create pipeline including data processing and the model. Export the pipeline as a pickle file.

<https://stats.stackexchange.com/questions/400242/model-deployment-export-scikit-learn-pipeline-or-model-only>

KNN Vs K-Means

KNN: Supervised, K=Nbr of neighbours used to classify a data element. Does not need training. If k=1 then error = 0. How to choose K=Plot between K and test accuracy.

K-Means: Unsupervised, K=Nbr of clusters. Randomly assign values to K nbr of clusters. Calculate their centroids. Reassign nearest neighbour to the corresponding cluster. Keep doing that until there are no more changes or improvements. For K to choose, plot between K and distortion. Distortion = sum of Euclidian distance within the clusters.

KElbowVisualizer

<https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/>

Cosine Distance

In n-dimension space of words, let their frequency be their magnitude. The distance between two words in a space is their angle of their corresponding vectors.

<http://blog.christianperone.com/2011/09/machine-learning-text-feature-extraction-tf-idf-part-i/>

Entity Embedding:

Embedding takes each label encoded value and assigns **latent factors** of fixed length in multi-dimensional space. The values of those latent factors are updated in DL training. Rather than using the list of latent factors, its corresponding index is used for look up. This is computationally effective. Also, embedding helps to provide the distance between two categories represented by their vectors in multi-dimensional space

Choosing embedding size: There is no fixed approach however Jeremy suggested an optimal number to use min(nbr\_of\_unique\_cats/2, 50). We use the cardinality of each variable (that is, its number of unique values) to decide how large to make its embeddings

For each category column train keras DL to get the

<https://towardsdatascience.com/deep-learning-4-embedding-layers-f9a02d55ac12>

Hyper-parameter tuning:

Grid Search – Tries all possible combinations

Random Search – Tries on random combinations

Bayesian Optimization – Use hyperopt package – suggest, trial and fmin

Parfit

Random Forest Optimization:

Regularization:

Reduce max\_depth

Bias and Variance trade off:

* + - Balance between overfitting and underfitting

XGBOOST

<https://stats.stackexchange.com/questions/354484/why-does-xgboost-have-a-learning-rate>

<https://www.kaggle.com/sudalairajkumar/xgb-starter-in-python>

<https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d>

Optimization:

N\_estimators = num\_boost\_round

early\_stopping\_rounds = early stopping (returns the final model instead of the best one)

Best\_ntree\_limit parameter can be used during prediction to isolate the best trained model, especially when early\_stopping\_rounds is set

Best\_iteration is just one lower than

GridsearchCV to use best\_ntree\_limit as well

Max\_depth: 3, 5

Learning\_rate: 0.03-0.05 to start with and then reduce to 0.01 or less

Reg\_lambda, reg\_alpha: for l2 and l1 respectively. High for lambda and low (0) for alpha

Customer learning rate reduction function in callbacks function

Gamma: to reduce overfitting by pruning

<https://medium.com/data-design/xgboost-hi-im-gamma-what-can-i-do-for-you-and-the-tuning-of-regularization-a42ea17e6ab6>

<https://medium.com/@gabrieltseng/gradient-boosting-and-xgboost-c306c1bcfaf5>

Interpretability:

Xgboost.plot\_importance(model, importance\_type = weight or cover or gain)

Eli5 package – show\_prediction to explain for each input. Permutation\_importance to explain prediction power of each feature by calculating how much score reduces when the feature is absent.

Pdpbox – plot\_pdp is used for partial dependence. Nb\_clusters for more number of rows for pd. Moreover effect of interaction by pdp\_interact\_plot

Finding interactions between features (h statistic for categorical and stata12 for continous): <https://pypi.org/project/sklearn-gbmi/>

<http://mathworld.wolfram.com/h-Statistic.html>

Skater

Lime

SHAP

<https://www.kaggle.com/datacog314/tutorial-machine-learning-interpretability/comments>

<https://towardsdatascience.com/introducing-pdpbox-2aa820afd312>

<https://towardsdatascience.com/explainable-artificial-intelligence-part-2-model-interpretation-strategies-75d4afa6b739>

<https://towardsdatascience.com/interpretable-machine-learning-with-xgboost-9ec80d148d27>

Although I presented gradient boosting as a regression model, it’s also very effective as a classification and ranking model. As long as you have a differentiable loss function for the algorithm to minimize, you’re good to go. The logistic function is typically used for binary classification and the softmax function is often used for multi-class classification

<http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/>

Clustering:

k-means:

sklearn.cluster.kmeans

standardize the data (so that the scale of distance is same)

treat missing values (this helps in running elkan algorithm that is more efficient in place of Llyods algo)

init= kmeans++ (initializes distant cluster means for better result)

n\_init = more than 100 (to run with different centroids for best model)

samples\_weights

Scree plot between nbr of clusters and distortion/inertia. This creates elbow diagram. However this is not enough. CH index (calinski harabasz ) is ratio between between clusters dispersion and within cluster dispersion. K with highest CH index is chosen.

Silhouette score (how close in a cluster is to another cluster: -1 to 1) for each cluster can be found as well. Higher the better.

Visualize using pandas parallel\_coordinates

PCA before the clustering can be done as well.

Agglomerative Hierarchical –

from scipy.cluster.hierarchy import dendrogram, linkage

It repeatedly links pairs of clusters until each data object is included in the hierarchy

Considers each data object as a cluster. Finds a pair of closest clusters and merges them. Keeps happening until one cluster is made on top.

We can see that the largest vertical distance without any horizontal line passing through it is represented by blue line. So we draw a new horizontal red line that passes through the blue line. Since it crosses the blue line at two points, therefore the number of clusters will be 2.

Basically the horizontal line is a threshold, which defines the minimum distance required to be a separate cluster.

Consider each row as a data object that has p dimensions (p = number of features)

Distance between each pair of data objects is a measure for cluster formation. The distance can be Euclidian or manhettan. Since the objects can be of different caliberation, those are normalized first. Moreover since the distance is always +ve, it is not really necessary to use Euclidean distance.

Non-hierarchical vs Hierarchical

Non-hierarchical – relationship between clusters is undetermined. With every run, the result might change because of centroid initialization.

Hierarchical – Keeps pairing and merging pair of closest clusters. Result is fixed.

<http://www.cs.ust.hk/~qyang/Teaching/537/Papers/huang98extensions.pdf>

<http://www.stat.cmu.edu/~ryantibs/datamining/lectures/06-clus3.pdf>

<https://stats.stackexchange.com/questions/134842/outlier-detection-with-data-which-has-categorical-and-numeric-variables-with-r>

<https://medium.com/datadriveninvestor/unsupervised-learning-with-python-k-means-and-hierarchical-clustering-f36ceeec919c>

<http://www2.cs.uregina.ca/~dbd/cs831/notes/clustering/clustering.html>

<https://www.daylight.com/meetings/mug96/barnard/E-MUG95.html>

For categorical data:

k-modes: K-modes algorithm distance is measured by the number of common categorical attributes shared by the two data points.

For both numeric and categorical data:

from kmodes.kprototypes import KPrototypes

Normalize vs Standardize

Normalize: Fits data between 0 and 1

Standardize: Makes mean 0 and standard deviation of 1

<https://www.statisticshowto.datasciencecentral.com/normalized/>

Correlation vs Covariance

Covariance denotes the direction of linear relationship between two variables. It is sum of cross products of two variables. Summation of (x - xmean)\*(y-ymean).

Correlation denotes both direction and strength between two variables. It is covariance divided by product of standard deviations of each variables. Values range from -1 to +1

<https://towardsdatascience.com/let-us-understand-the-correlation-matrix-and-covariance-matrix-d42e6b643c22>

ngrams

A group of adjacent words or characters in a string. Length of group=n and it considers boundary as well

<https://stackoverflow.com/questions/18193253/what-exactly-is-an-n-gram>

What if categorical values present in test but missing in train:

1. One-hot all the values from both train and test. If for the one missing in train, model will not give any weight though
2. Converting category to a function of numeric value, if at all possible. Like converting city to function of latitude and longitude
3. If not so important and very less number of entries, then can be moved to “other”

Large Number of categorical values:

1. Use categorical\_encoder package to do target\_encoder or hashing
2. Convert cat2vec using word2vec
3. Create Embedding
4. If cardinality < 7, then one-hot, else label encoding. If ordinal then also label encoding

<https://datascience.stackexchange.com/questions/37480/how-can-i-do-classification-with-categorical-data-which-is-not-fixed>

Gradient descent

<http://ruder.io/optimizing-gradient-descent/>

batch gradient descent – slow and guaranteed to reach local minima

stochastic – fast but updates on loss function fluctuates much and hence reaching local minima is a problem

mini batch – best of both

Optimizers:

SGD: with momentum

Adam: combines both Adagrad and RMSProp

<https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>

Time Series

ARIMA (autoregressive integrated moving average)

Box-jenkins method

P = nbr of lag order (check autoregression graph and decide)

D = degree of differentiation (to remove seasonality)

Q = window of moving average

Residuals should follow normal distribution

<https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/>

Noise in tabular data can be of three types:

* Anomalies in certain data items (Noise 1: certain anomalies in features & target)
  + Outlier detection & treatment: either remove the records or put upper and lower ceiling
  + <https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba>
  + <https://towardsdatascience.com/outlier-detection-and-treatment-a-beginners-guide-c44af0699754>
  + If data distribution is normal then check if <> 3 sigma else inter quartile range
* Features that don’t help in explaining the target (Noise 2: irrelevant/weak features)
  + Filter method: Pearson’s Correlation, Anova, LDA, Chi-Square (These methods don’t identify or deal with multicollinearity, we need to figure that out separately)
  + Wrapper method: Forward selection, Backward elimination, Recursive elimination
    - <https://github.com/scikit-learn-contrib/boruta_py>

(It uses Random Forest Classifier and Cross Validation)

(It uses Ensembling technique. The loss in error function at the split point can be calculated as SSE and the same as Gini Score (considers density mas function to calculate the impurity score in sample) in Classification to find the importance of a variable)

<https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/>

* + This method makes use of filter & wrapper method, it is implemented using algos which have its own built-in feature selection methods – Lasso and Ridge
* Records which don’t follow the form or relation which rest of the records do (Noise 3: noisy records)
  + K-fold validation
  + Manual method
  + Unsupervised Methods (Anomaly Detection)
    - Density-based anomaly detection
      * This method assumes normal data points occur around a dense neighborhood and abnormalities are far away. i.e. kNN & LOF based methods
    - Clustering-based anomaly detection
      * Using clustering technique, we can analyse the clusters to analyse which has noise. Data instances falling outside the clusters can be marked as anomalies. i.e. k-Means clustering
    - SVM-based anomaly detection
      * This technique uses SVM to learn the soft boundary in the training set and tune on validation set to identify anomalies. In this approach, the need of large samples by the previous approach is reduced by using Support Vector Machine while maintaining the high quality of clustering-based anomaly detection methods. i.e. One-class SVM
    - Autoencoder-based anomaly detection
      * Auto-encoders are used in deep learning for unsupervised learning, we can use them for anomaly detection to identify noisy data-set. These methods are advanced and outperforms traditional anomaly detection methods. i.e. Variational Autoencoder based Anomaly Detection using Reconstruction Probability.

Benefits of identifying & treating noise in data:

* enables the DS algorithm to train faster
* reduces the complexity of a model and makes it easier to interpret
* improves the accuracy of a model if the right subset is chosen
* reduces overfitting

ref:

<https://medium.com/analytics-vidhya/dealing-with-noisy-data-in-data-science-e177a4e32621>

<https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/>

Improve the performance as the predictive models:

* Working on data (further data cleaning, feature engineering, feature selection, regularization etc)
* Working on models (appropriate model & evaluation metrics selection, re-sampling techniques, cross-validation etc)
* Tuning the models (optimizing hyper-parameters, random & grid search etc)
* Ensembling the models (bagging, boosting, stacking etc)
  + Basic:
    - Max Voting
      * Used for classification with mode of predicted values from all models
    - Average
      * Used for regression with average of predicted values from all models
    - Weighted Average
      * Used for regression. Taking average of weighted predictions
  + Advanced:
    - Stacking
      * Predictions on K-folds and test set from the base models are used by another model to make final prediction
    - Blending
      * Similar to Stacking but instead of K-folds, hold out (validation) set is used
    - Bagging (Bootstrap Aggregation)
      * Taking multiple samples from your training dataset (with replacement) and training a model for each sample. The final output prediction is averaged across the predictions of all of the sub-models
      * Algorithm with high variance are preferred as base algorithm such as Decision Tree, KNN with low K value or any other algorithm with high variance
      * Bagged Models: Decision Trees, Random Forest, Extra Trees
    - Boosting
      * Boosting ensemble algorithms creates a sequence of models that attempt to correct the mistakes of the models before them in the sequence.
      * Boosting algorithms – AdaBoost, Stochastic Gradient Boosting (Gradient Boosting Machines), XGBOOST, LightGBM, Catboost
      * <https://medium.com/greyatom/boosting-ce84639a805d>
      * <https://www.python-course.eu/Boosting.php>

Ref:

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

<https://machinelearningmastery.com/ensemble-machine-learning-algorithms-python-scikit-learn/>

Deep Learning

Tuning LSTM

Dropout. Slow down learning with regularization methods like dropout on the recurrent LSTM connections.

Layers. Explore additional hierarchical learning capacity by adding more layers and varied numbers of neurons in each layer.

Regularization. Explore how weight regularization, such as L1 and L2, can be used to slow down learning and overfitting of the network on some configurations.

Optimization Algorithm. Explore the use of alternate optimization algorithms, such as classical gradient descent, to see if specific configurations to speed up or slow down learning can lead to benefits.

Loss Function. Explore the use of alternative loss functions to see if these can be used to lift performance.

Features and Timesteps. Explore the use of lag observations as input features and input time steps of the feature to see if their presence as input can improve learning and/or predictive capability of the model.

Larger Batch Size. Explore larger batch sizes than 4, perhaps requiring further manipulation of the size of the training and test datasets

How to evaluate Deep Learning Model:

Since DLs are stochastic in nature, that is changing the output in every run, taking the average of the result will not suffice. It is recommended to take the mean as well as the standard deviation of the results from repeated testings on the same data.

<https://machinelearningmastery.com/evaluate-skill-deep-learning-models/>

RNN

A slightly more dramatic variation on the LSTM is the Gated Recurrent Unit, or GRU, introduced by Cho, et al. (2014). It combines the forget and input gates into a single “update gate.”

Vanishing / Exploding Gradients

Vanishing:

During gradient descent, as it backprop from the final layer back to the first layer, gradient values are multiplied by the weight matrix on each step, and thus the gradient can decrease exponentially quickly to zero. As a result, the network cannot learn the parameters effectively.

Solutions:

Use LSTM/GRU in the sequential model

Modern RNN like LSTM/GRU introduced the concepts of "gates" to artificially retain those long-term memories

In GRU, there are two gates - update and relevant

In LSTM, there are three gates - input, output, forget

Use Residual network

The idea of the residual network is to allow direct backprop to earlier layers through a "shortcut" or "skip connection

Use ReLu activation instead of Sigmoid/Tanh

Weight Initialization

Keras default weight initializer is glorot\_uniform aka. Xavier uniform initializer. Default bias initializer is “zeros”. So we should be good to go by default.

Exploding:

During training, it causes the model's parameter to grow so large so that even a very tiny amount change in the input can cause a great update in later layers' output.

We can spot the issue by simply observing the value of layer weights. Sometimes it overflows and the value becomes NaN

Solutions:

Gradient clipping for Exploding gradients

As this name suggests, gradient clipping clips parameters' gradients during backprop by a maximum value or maximum norm

Apply Regularization like L2 norm for Exploding gradients

L2 norm applies "weight decay" in the cost function of the network

How correlated features affect the prediction:

It might not affect anything. Other than adding more storage and creating necessity for more computational power.

<https://datascience.stackexchange.com/questions/24452/in-supervised-learning-why-is-it-bad-to-have-correlated-features>

Random Forest: As we remove one feature, it model select updated important features.

PCA: It removes highly correlated features when SVD is applied for Eigenvalue calculation

In linear regression, it does matter as it would result in high variance in the distribution of coefficient values and changing the value of R square.

<https://stats.stackexchange.com/questions/86269/what-is-the-effect-of-having-correlated-predictors-in-a-multiple-regression-mode>