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

Ref:

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

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

Big amount of data:

- When we need to deal with big data for data science, we should break the problem into two parts, first to build an understanding & data science baseline on random samples (to iterate & learn quickly) and then to work on scaling that pipeline to handle big data

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 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/

Regression:

Assumptions and Solutions:

- 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
    - 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

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/

Ridge

Good for highly multi correlated data
Uses L2 regularization
Does not shrink the coefficients to zero, hence no feature selection

Lasso
Good for highly multi correlated data
Uses L1 regularization
Shrinks the coefficients to zero, hence does help with feature selection

Basic Metrics To Assess 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

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.

Algorithms for split:
Gini index
Chi square
Information gain
Reduction in Variance

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/

What is pruning in Decision Trees:
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.

Parameter tuning for xgboost:
Step1: Fix the values to start with
max_depth = 5
min_child_weight = 1

gamma = 0
subsample, colsample_bytree = 0.8
scale_pos_weight = 1
Step 2: Tune max_depth and min_child_weight
Step 3: Tune gamma
Step 4: Tune subsample and colsample_bytree
Step 5: Tuning Regularization Parameters
Step 6: Reducing Learning Rate

SVM:
Kernels:
Liner
Gaussian
RBF
Anova
https://stackoverflow.com/questions/33778297/support-vector-machine-kernel-types

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

Naïve Bayes model:
Assumption: There should not be any correlation between two features in the input
Note: It is useful for huge dataset

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.

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

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

Difference Between R square and adjusted R square

The only drawback of R^2 is that if new predictors (X) are added to our model, R^2 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".

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

List of Linear Models

Simple Linear Regression

Logistic Regression

Lasso Regression (L1)

Ridge Regression (L2)

Elastic Net

Naïve Bayes

Learning Rate

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

Classification: Evaluation metric (For both binary and multiclass and multi label)
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

Precision: TP/(TP+FP)
Cannot afford any false alarm like irrelevant result in search engine

F1 score: 2 * (recall * precision) / (recall + precision)
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.

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

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.

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)

Penalize more for wrong classification for certain classes

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/

Def of P value

SQL
PySpark
Hive
In depth about all algos
Check algos in sklearn

LDA
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