**AI Exam Process Steps with Functions:**

1. **Initial steps**
   1. **Load your dataset**
   2. **Examine it and get rid of anything that doesn’t apply**
      1. *Categoricals(df)*
      2. *examine\_df(df)*
      3. *distribution\_plot(df,column\_name)*
      4. *joint\_plot(df,x,y)*
      5. *pair\_plot(df)*
   3. **Split the dataset into the input and output variables**
      1. *separate\_components(df, column\_of\_y)*
2. **Pre-processing**

**Visulization Inspiration:** https://www.kaggle.com/bakosab666/heart-disease-visualization-and-predicting

Can be either done using first fit() then transform() function or fit\_transform()

* 1. **Rescale data(Normalization)**

Important for optimization algorithms that use gradient descent and regressions that weigh inputs(regression/neural networks or with clustering – k-means)

* + 1. *rescale (X)*
  1. **Standardize data (1 to 0)**

Change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values

Assumes Gaussian distribution but different means and standard deviation. 0 means mean and 1 means standard distribution. Most relevant for linear, log regreassion and LDA(linear discriminant analysis)

* + 1. *standardize(X)*
  1. **Normalize data (Each row equals 1)**

Normalize by row so that each row length 1. Useful for algorithms that weigh input values as a whole such as Neural Networks and distance algorithms(K-nn)

* + 1. *normalize(X)*
  1. **Binarize data (on binary threshold)**

Transform data based on binary threshold. Used when adding a new feature often.

* + 1. *binarize(X)*
  1. **Label Encoding**

Changing name of column

* + 1. *encode(df, name\_of\_column, new\_name)*
  1. **Dummy encoding**

Create dummy variables for each class to avoid order when in reality only encoding

* + 1. *get\_dummies(df, column\_name*
  1. **Cutting Category**

Create category using cutoffs and label to turn numeric continuous to categorical

* + 1. *cut\_category(column,arr\_of\_cutoff,labels,new\_col\_name, position)*

1. **Feature Selection**

Quality of model determined by data features. Select those features that contribute most to the variable of interest. It reduces overfitting through reducing redundancy. Improves accuracy by creating a more accurate model. Reduces time needed to train by excluding irrelevant features.

Recursive Feature selection already fits to a model which might make sense if we use it in the next step anyways.

https://www.datacamp.com/community/tutorials/feature-selection-python

* 1. **Univariate Selection**

Uses statistical test mostly with KBest to determine most relevant attributes using statistical test. Determine strongest relationship with output variable.

* + 1. *univariate\_chi(x, y, df, target\_var, k=4(default))*
  1. **Recursive feature elimination**

Recursively removes attributes and builds models with remaining. Model accuracy as guidance on which combination of attributes contributes the most.

* + 1. *recursive\_elimination(x, y, df, target\_var, k=3(default))*
  1. **PCA – Principal Component Analysis**

Data reduction method with linear algebra through compressing several dimensions into principal components. Cannot explain them once comporessed.

Method combines highly correlated variables together to form smaller number of an artificial set of variables called “principal components” that account for most variance in the data.

* + 1. *pca(x, k=3(default))*
  1. **Tree Classifier**

Can be used to estimate importance of feature to refine the model and to make decision beteen explainability and accuracy. Many trees with variations to get accurate result. Aggregates the results of multiple de-correlated decision trees collected in a “forest” to output it’s classification result

* + 1. *Extra\_trees(x,y,df,target\_var, estimators=100(default))*

1. **Model Evaluation**

Two reasons for evaluation:

Decide which interventions are effective

Get concrete knowledge on how accurate they are and to what extent we can trust them

We cannot use data we used for training the model to assess it. Two approaches to mitigate it – 1) use Train-Test Split works well with a lot of data 2) sampling techniques to augment the amount of data available

1. **Train and test set**

Split dataset into 70-30 or 80-20 split. Simple, fast and good for large datasets. We have to ensure variance is similar. If not, might not be good predictor.

* 1. *test\_split(x,y,test\_size, seed=7(default))*

1. **K-fold Cross-validation**

Used to reduce variance of using test-train-split. Divide data into folds and train multiple times. Dataset must be large to accommodate process. Takes stratified kfold to ensure representative representation of outcome variables and avoid overrepresentation of non-representative datasets

* 1. *k\_fold\_cross\_val(x,y,splits=10(default))*

1. **Leave one-out cross-validation**

As many folds as point. Prediciton will be done for each point separately. Standard deviation has more variance than with kfold and more computational expense

* 1. *leave\_one\_out(x,y)*

1. **Repeated random test-train splits**

Repeatedly takes train-test split. Results might look promising but are not realistic as same data is used over and over again.

* 1. *repeated\_test\_train(x,y, test\_size,repetitions=10)*

1. **Performance Metrics**

Metric against which model will be tuned, hyperparameters will be selected and algorithm will be evaluated

Classification Problem: Attributes are numeric and binary classification problem

Regression problem: Regression problems as continuous relationship assessement

**Classification Metrics:**

* 1. **Accuracy**

Ratio of (correct predictions/all predictions). Only suitable if classes are balanced and errors in each similarly important (wrong cancer diagnosis vs. no diagnosis of ill)

* + 1. *precision\_recall(x,y,test\_size)*
  1. **Logarithmic Loss**

Evaluates predictions of probabilities on a scalar between 0 and 1 as confidence measure.

* + 1. *k\_fold\_cross\_val(x,y, scoring = ‘neg\_log\_loss’)*
  1. **Area under ROC curve**

Ability of model to discriminate between positive and negative classes. AUC of 1 means perfect model: Shows performance of classification model at all classification thresholds.  
Sensitivity: True positive (positive values correctly classified)  
Specificity: True negative (negative values correctly classified)

* + 1. *area\_under\_roc(x,y,test\_size)*
  1. **Confusion Matrix**

Presentation of accuracy of the model in its four classes(TP, TN, FP,FN).

* + 1. *con\_matrix(x,y,test\_size=0.3)*
  1. **Classification Report**

Report to summarize many of the accuracy measures  
a) Precision = TP/TP+FP  
b) Recall = TP/TP+FN  
c) F1-Score = 2 x Precision x Recall / Precision + Recall (good for uneven distribution of classes)  
d) Support – number of elements of each class in Y-Teset  
e) Averages

* + 1. *class\_report(x,y, test\_size=0.3)*

**Regression Metrics:**

1. **Mean Absolute Error (MAE)**

Sum of absolute differences between values and prediction. 0 is perfect fit.

* 1. *mean\_abs\_error(X,Y)*

1. **Mean Square Error(MSE)**

Square value of absolute differences. Maximize score.

* 1. *mean\_squared\_error(X,Y)*

1. **R2**

Explained variance by regression

* 1. *r\_2(X,Y)*

1. **Clustering**
   * 1. Choose the number K of clusters
     2. Select a random K points (the initial centroids)
     3. Assign each datapoint to nearest centroid
     4. Compute and place new centroid of each cluster(move to mean)
     5. Assign each datapoint to each cluster

Unsupervised learning technique. Not one single solution. Clusters depend on point of view and question we try to answer. Finding best number of clusters, checking goodness and multidimensional clusters as some of the challenges we have. Additionally, as we are dealing with distances, all variables have to be rescaled from 0 to 1. Because Euclidian distances are skewed towards larger dataset.

In some cases, clustering coincides with classification. When we know class, we can apply measures such as accuracy. Otherwise, it is hard to assess accuracy.

* 1. **Rescale and fit to columns**

Columns have to be rescaled and with column title

* + 1. *scale\_cluster\_df(x)*
  1. **Elbow Method**

Used to identify optimal number of clusters that minimize intra-cluster distance.

* + 1. *elbow(x,y)*
  1. **Silhouette Visualization**

Checking goodness of clusters. Measures mean intra-cluster distance relative to nearest clusters providing an appreciation of compactness.

* + 1. *Silhouette\_visual(x\_scaled)*
  1. **Inter-cluster distance**

The further apart centroids are the better

* + 1. *inter\_cluster\_dist(x\_scaled)*
  1. **Accuracy clustering**We can determine accuracy of cluster if we know classes.
     1. *accuracy\_clustering(x\_scaled, y)*
  2. **Silhouette score**Measure of how similar object is to own cluster compared to other clusters. Ranges from +1 to -1 and high value indicates that object is well matched to own cluster. If low or negative, too many or too few clusters chosen.
     1. *silh\_sco(x\_scaled, model)*
  3. **Centroid rescaling for meaning**

Centroids can only be interpreted if we rescale them from 0 to 1 range to their original ranges. On this basis we can interpret them.

* + 1. *centroid\_meaning(model, x, x\_scaled)*
  1. **Agglomerative Hierarchical Clustering**

Start with every point as one cluster and agglomerate with next closest point to a cluster until entire data set is a cluster

* + 1. *aggl\_cluster(x\_scaled,clusters,y)*

1. **Classification**

X is scaled using standard scaler and minmax scaler. Normaly perform better with classification.

* 1. **Logistic Regression**

Assumes Gaussian distribution for numberic input variables and can solve binary and multi-class classification problems.

* + 1. *log\_regression(x,y,splits)*
  1. **LDA – Linear Discriminant Analysis**

Assumes that data is gaussian and each variable shaped like a bell curve when plotted. Each attribute has same variance. Estimates the propability that new set of inputs belongs to each class. Should be gaussian (using transformations (e.g. log and root for exponential distributions and Box-Cox for skewed distributions)

* + 1. *lda(x,y,splits)*
  1. **k-nn k-Nearest Neighbours**

Non-linear machine learning algor^ithm that uses distance metrics to find most similar k-elements. It takes mean outcome of neighbors as prediction

Can choose different metrics for calculation of distance. Default is Minikowski (equal to Euclidean (with p=2). Can be easily transformed to Manhatten distance with p=1  
Choosing and plotting beset k-value: https://medium.com/@kbrook10/day-11-machine-learning-using-knn-k-nearest-neighbors-with-scikit-learn-350c3a1402e6

* + 1. *k\_nn\_k\_nearest(x,y,splits)*
  1. **Naïve Bayes**

Class labels are represented by vector of features. Each feature is considered independent of others (why it is called naïve). Probabilities are calculated following Bayesian approach. Works quite well, especially on small data samples.

Features are assumed to be completely independent  
Each feature is given the same weight and none of them is irrelevant and all contribute equally.

Each distribution can be independently estimated as one-dimensional, reducing the curse of dimensionality

* + 1. *naive\_bayes(x,y,splits)*
  1. **CART – Classification and Regression Trees (Humble decision tree)**

Builds binary tree from data where splits are chosen greedly evaluating all attributes to minimize a cost function. This is the base for random forests and more sophisticated algorithms. Split with best cost function is selected.

Gini index signals proportion of training instances with class k in rectangle of interest

Traverses tree based on input. By just following path on each dimension.

Uninformative variables will be automatically excluded.

* + 1. *decision\_tree(x,y,splits) / decision\_tree\_vis(df,model,x,y,zero\_class, one\_class)*
  1. **Support Vector Machines**

Seeks line that seperates two classes best. Data instances that are closest to this line are called support vectors. You can change the kernel function to use, default is radial basis function.

* + 1. *support\_vector\_machine(x,y,splits)*

1. **Basic Regression**
   1. **Linear Regression**

Assumes Gaussian distribution with all variables relevant. Variables should not be highly correlated (multicollinearity).

* + 1. *evaluate\_regressors(X, y, n\_splits=10, shuffle=True, random\_state=0, scoring='neg\_mean\_squared\_error')*
  1. **Ridge Regression**

Modern linear regression more resilient towards outliers. Loss function modified to minimize complexity of model measured as sum squared value of coefficient values. Can be used also when multicollinearity present. Starts with slightly worse initial prediction to achieve more accuracy over long-run. Adds small bias trhough penalty. Tries to minimize sum of squared residuals + ridge penalty

* 1. **Lasso Regression**

Also aims at reducing weight of outliers. Loss function modified to measure complexity as sum absolute value of coefficient values and penalty.

* 1. **ElasticNet Regression**

Combines Ridge ad Lasso. Penalizes model using both L1 and L2 norm

* 1. **K-nn k-Nearest Neighbours**

Uses distance metric to find most similar k-elements by taking mean or median outcome of neighbours as predictions. We can choose different metric for calculating the distance, either Minkowski or Manhattten.

* 1. **CART – Classification and Regression Trees**

Builds binary tree from data where splits are chosen greedly evaluating all attributes to minimize a cost function. Mean squared error as default specified parameter.

* 1. **Support Vector Machines**

Seeks line that best seperates two classes. Instances that are closest to this line are better separating the classes. Can change kernel function to use. Radial basis as default.

1. **Ensembles**

Two main ideas: 1. Combining weak learners we can get a strong learner. 2. Reduce and overcome overfitting by implementing combinations of same learner to reduce variance and increase performance

* 1. **Bagging**

Build multiple models, typically same type, from different subsamples of a dataset and combining them with the mean. Is the contraction of bootstrapping and aggregation. Random Forest as extension by trying to reduce correlation between individual classifiers. Random number of features in each split. Extra trees, each tree trained using whole sample and topdown split randomized.

* 1. **Boosting**

Build model incrementally where each iteration tries to fix the errors of tprevious one. Weighting instances of dataset acc. To their difficulty to classify and using this to pay more or less attention to each instance when constructing subsequent models. Combines weak learners to make classification. Some stump get more weighting. New stumps made using previous mistakes.

* + 1. AdaBoost
  1. **Voting**

Multiple models, typically of different types and a procedure to combine their predictions. First, create all models using training dataset and when predicting take average of predictions of submodel.

* 1. **Feature Importance**

1. **Hyperparameter Tuning**

Once we have selected most appropriate algorithm, we can still improve its performance by selecting most appropriate parameters. Hyperparameters are not part of model itself but parameters of algorithm.

* 1. **Grid Search Parameter Tuning**

Provide a grid filled with potential options for parameters and instruct system to try them all. Once you get results you can pick up the one that suits you best.

* 1. **Random Search Parameter Tuning**

You don’t provide parameters but system searches randomly.

1. **XGBoost**

Implementation of Gradient Boost Machines for boosted tree algorithms. Aims to improve the solution by building models that correct errors of previous models. These models are added sequentially until no more improvements can be made. Supports three main forms:

1 Gradient boosting algorithm including the learning rate

2 Stochastic Gradient Boosting: sub-sampling at the row, column and column per split levels

3 Regularized Gradient Boosting using L1 and L2 regularization see Ridge and lasso

XGBoos is fast, memory efficient and highly accurate