Chapter 1 – Data Preparation

Please follow:
https://rb.gy/n6yaja
Chapter 2 – Regression

Please follow:
https://rb.gy/ub4nxu
Chapter 3 – Classification

Please follow:
https://rb.gy/whnkbh
Deeper dive into Decision Trees

- Tree structured Classifier, used for Classification problems & Regression
- Branches, Decision Nodes and Leaf Nodes

Measure used to split a node:
To reduce Classification error
Gini:
- measure of impurity in a node
- Information gain
Entropy:
- Measure of Disorder

Regression
Residual squared error

- Editing of Income data
- Binary Classification – No_Change/Change
- Orange – No_Change, Blue - Change

when AnticPay_0.0<=0.5 and Nins>1.5 and NetPay<=4724.66992188 and IncTax<=0.5 and DVUsHr<=35.5 and Totus1<=22.25

Dubai EXPO 2020
# initialize the model
net_pay_Tree = tree.DecisionTreeClassifier(min_samples_leaf = 20)

# train the model
net_pay_Tree = net_pay_Tree.fit(df_pre_edit_train.drop(['Change'],axis=1),df_pre_edit_train['Change'])

# run the prediction on the test data and place the result into a new data frame df_net_pay_pred_test_proba
# this will hold the probability values of the prediction that a test case needs changing
df_net_pay_pred_test_proba = net_pay_Tree.predict_proba(df_pre_edit_test.drop(['Change'],axis=1))[:,1]

# create a binary data frame where a '1' indicates a probability of > 0.5, Threshold = 0.5
df_net_pay_pred_test_binary = df_net_pay_pred_test_proba > 0.5
Random Forest

- Single trees are weak classifiers:
  - Slight change of data $\rightarrow$ very different tree
  - Different tree $\rightarrow$ different prediction

- Ensemble of many trees $\rightarrow$ Random Forest
  - Random selection of features for each tree $\rightarrow$ every feature can show its decision making power
  - Bagging (Bootstrap Aggregation) – Random Sample with Replacement of Training Data
    $\rightarrow$ each sample can have zero, one or more copies of the training records
    $\rightarrow$ each tree is trained with different data sets, but all have same size
  - Reduces dependency on training data $\rightarrow$ more accurate prediction
  - More accurate Feature Importance
  - Each tree ‘votes’ on the prediction $\rightarrow$ prediction score
Bootstrap Aggregation or Bagging – Random samples with replacement

Random Feature Selection
Managing Complexity of Tree based Models

**Over-fitting**
- Boundary not well defined, complex rules
- Very good Training data predictions

**Under-fitting**
- Well defined Boundaries, simple rules
- Not so good predictions
Hyperparameters in Random Forest

net_pay_Tree_orig = RandomForestClassifier

    (bootstrap = True, # bagging
criterion = 'gini', # gini measure to split nodes
max_depth = 40, # depth of tree
max_features = 'sqrt', # number of features for each split
max_leaf_nodes = 400, # grow trees with this number of leaf nodes
min_samples_leaf = 5, # minimum of records in each leaf
n_estimators = 1000, # number of trees
n_jobs = -1 # number of processors used, all if -1)
Hyperparameters tuning with GridSearch

```python
parameter_grid = [
    {
        "bootstrap" : [True],
        "criterion" : ["gini"],
        "max_depth" : [40,45,50],
        "max_features" : ["sqrt"],
        "max_leaf_nodes" : [180,240,280],
        "min_samples_leaf" : [2,4,8,12,18],
        "n_estimators" : [165,175,200],
        "n_jobs" : [-1]
    }
]

net_pay_Tree = model_selection.GridSearchCV(ensemble.RandomForestClassifier(),
                                             parameter_grid,
                                             scoring = "f1",
                                             cv = 5)

# 135 RandomForest will be trained
net_pay_Tree.best_params  # prints parameters for best RandomForest based on parameter_grid and scoring metric
```
Chapter 4 – Dimension Reduction

Please follow:

Chapter 5 – Clustering

Please follow: