* Train test and tx ty split using sklearn
* t-sne 2d visualization for approved denied cases seperability. I.e. representation of data on 2d plot
* count plot with target-var hue
* feature engg. – creation, selection and extraction
* how to find correlation between categorical features – pair plot
* one-hot encode of all cat variables using sklearn (which create sparse scipy array instead of numpy array– uses less memory). We first must convert into label encode to use this feature
  + le = LabelEncoder()
  + train\_le = le.fit\_transform(train)
  + ohe = OneHotEncoder()
  + train\_ohe = ohe.fit\_transform(train\_le.reshape(-1,1))
  + Instead of (Label -> 1hot) use (LabelBinarizer(sparse\_output = True))
* Distribution of numeric features and scikit scaling (min-max scaling and standardization)
* Cross validation score with mean and sd in function
* Test-out all the classifier models with cv score -> shortlist 5-6 best promising models -> fine tune them -> come out with the best hyper-parameter grid -> compare -> select uncorrelated classifiers -> ensemble (**remember: your model is yet not exposed to the test set, you are evaluating only on cv**)
* Use stratified k fold = 5 for faster processing
* Feynman-understanding of these finalized models and their hyperparameters
* When you finally evaluate the fine tune model performance on test set, it might be slightly lower because model is tuned on cross-validation. But resist the temptation to tune your model on test set.
* Adjust threshold for best approved vs. Denied accuracy, I guess using pr curve and roc curve
  + Y\_scores = cross\_val\_predict(sgd\_clf,x\_train,y\_train, cv=5, method=’decision\_function’)
  + Y\_predict\_probas = cros\_val\_predict(rf, x\_train, y\_train, cv=5, method=’predict\_proba’)
  + P,r,t = precision\_recall\_curve(y\_train, y\_scores) -> Plot
  + Fpr, tpr, t = roc\_curve(y\_train, y\_scores)
* As a rule of thumb, u should use pr curve when u care more about false positives than false negatives and roc in vice-versa
* Diagnosing bias-variance problems with learning curves. Accuracy (y-axis) vs. No. of training samples (x-axis). Training accuracy curve, Validation accuracy curve and Baseline accuracy line
* DumbClassifier code
  + From sklearn import BaseEstimator
  + Class DumbClassifier (BaseEstimator):
  + Def fit(self, X, y=none):
  + Pass
  + Def predict (self, X):
  + Return np.ones((len(X), 1), dtype=bool)
* Use cross\_val\_predict instead of cross\_val\_score
  + Y\_train\_pred = cross\_val\_predict(sgd\_clf, x\_train, y\_train, cv=5)
  + Metrics.confusion\_matrix(y\_train, y\_train\_pred)
  + Metrics.precision\_score(y\_train, y\_train\_pred)
* Classifier list:
  + SGDClassifier
  + DumbClassifier
  + Logistic
  + Decision tree
  + Bagging classifiers sklearn
  + Boosting classifiers sklearn
  + SVM
  + MLP
* Check the unix machine config and if higher, process there
* Voting classifier proof-of-concept -> soft/hard vc of logistic, rf and linear svc
* Bagging oob\_score = True gives the idea of test accuracy -> bag\_clf.oob\_score\_
* Stacking – instead of using hard voting to aggregate predictions, why don’t we train the model to perform this aggregation. Different models predict, then a final classifier (meta learner a.k.a. blender) takes these predictions as input and makes the final predictions.
  + To train the blender, common approach is hold-out set
  + There is an open source implementation ‘brew’. Scikit yet not support stacking
* Work on function – calculate information value
* We don’t need feature extraction (a.k.a. dimensionality reduction) as such. Only feature selection for seeing feature importance and feature selection to reduce overfitting
* Keras neural net tuning and model training evaluation
* Clustering (a.k.a. unsupervised classification) – distance-based models and probabilistic models
  + K-means - db
  + DBSCAN -db
  + Agglomerative/ Heirarchical
  + Topological clustering – self organizing map (SOM) – wiremesh representation
  + Spectral
  + Mixture models - pm
  + Autoencoder – refer to Kaggle example titanic dataset
* From sklearn.cluster import AgglomerativeClustering, DBSCAN, KMeans
* From sklearn.cluster import MeanShift, estimate\_bandwidth, SpectralClustering
* From hdbscan import HDBSCAN
* Data type changes – nominal
* Autosklearn tryout in unix
* How to use Catboost, Xgboost and light boost algorithms
* In case of unsupervised – clustering, the performance metric is ‘silhouette coefficient’. This help you understand 2 things – cohesion (similarity b/w clusters) and separation.
* Use warm\_start = 1 during randomizedsearchcv. Next level is Bayesian optimization. Sequential model-based Global Optimization (SMBO). Sequential Model-based Algorithm Configuration (SMAC) is a great library that uses Bayesian optimization
  + From smack.tae.execute\_func import ExecuteTAFuncDict
  + From smac.scenario.scenario import Scenario
  + From smac.facade.smac\_facade import SMAC
* Experiments and building sklearn pipelines. You have to put everything in this pipeline format.
* Preprocess
  + Split train-test
  + Missing data – drop, impute with mode, separate as a category, impute with prediction
  + Data validation and correction
  + Data standardization and scaling
  + Encoding
  + Modelling fit and transform
  + Evaluation
* I think by insights generation through unsupervised learning, meaning is to find patterns in data. Along with clustering on subset of features, multi-variate EDA would also helpful.
* Clustering outcomes usages
  + Target variable %approval and %denied ratio in each cluster
  + Case/procedure similarities pattern. After clustering, find the reasons of why particular cases are in same cluster
* Association rule mining – a good technique for categorical data
* CART models comparison with n\_estimator changes – 50, 100, 500, 1000 and 5000
* [Kmodel library](https://medium.com/@davidmasse8/unsupervised-learning-for-categorical-data-dd7e497033ae) (PIP install KModes inside your env1 VM). [Paper](http://www.cs.ust.hk/~qyang/Teaching/537/Papers/huang98extensions.pdf)
* <https://scikit-learn.org/stable/unsupervised_learning.html>
* One strategy is to create principal components using MCA/CATPCA and then do clustering on that. A good [research paper](http://www.nada.kth.se/~ann/exjobb/sara_engardt.pdf) on clustering.
* How kagglers do EDA
  + Understand the problem. We'll look at each variable and do a philosophical analysis about their meaning and importance for this problem
* How RFE works?
  + One method for doing this automatically is the Recursive Feature Elimination method in Scikit-Learn. This accepts an estimator (one that either returns feature weights such as a linear regression, or feature importances such as a random forest) and a desired number of features. In then fits the model repeatedly on the data and iteratively removes the lowest importance features until the desired number of features is left. This means we have another arbitrary hyperparameter to use in out pipeline: the number of features to keep!
* Dimensionality reduction is one of the most popular techniques to remove noisy (i.e. irrelevant) and redundant features. Dimensionality reduction techniques can be categorized mainly into feature extraction and feature selection. Feature extraction approaches project features into a new feature space with lower dimensionality and the new constructed features are usually combinations of original features. Examples of feature extraction techniques include Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Canonical Correlation Analysis (CCA). On the other hand, the feature selection approaches aim to select a small subset of features that minimize redundancy and maximize relevance to the target such as the class labels in classification. Representative feature selection techniques include Information Gain, Relief, Fisher Score and Lasso.
* Pipelines are set up with the fit/transform/predict functionality, so you can fit a whole pipeline to the training data and transform to the test data, without having to do it individually for each thing you do. Super convenient, right?? Pipelines help you prevent data leakage in your test harness by ensuring that data preparation like standardization is constrained to each fold of your cross validation procedure.
* ### Visualization Theory
  + 1. Single Categorical Variable X (Count-Plot or Factor-Plot)
  + 2. Continuous Variables XY (Scatter-Plot or Pair-Plot)
  + 3. Single Continuous Variable Y (Box-Plot or Violin-Plot or KDE-Plot)