* Distribution of numeric features and scikit scaling (min-max scaling and standardization)
* 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**)
* 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)
* 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
* We don’t need feature extraction (a.k.a. dimensionality reduction) as such. Only feature selectionelimination for seeing feature importance and feature selection to reduce overfitting
* 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
* 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.
* Association rule mining – a good technique for categorical data
* CART models comparison with n\_estimator changes – 50, 100, 500, 1000 and 5000
* <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)
* Correlation analysis between 2 categorical variables – Chi2 test.
  + Null hypothesis: they are independent, Alternative hypothesis is that they are correlated in some way.
  + Correlation between independent variables should be low (no multicollinearity) and correlation between an independent and dependent variable should be high.
  + For categorical, we use lambda, Cramer V instead of Pearson as a metric.
* Different types of distance metrics
  + Sum of Absolute Distance
  + Sum of Squared Distance
  + Mean-Absolute Error
  + Euclidean Distance
  + Manhattan Distance
  + Chessboard Distance
  + Minkowski Distance
  + Canberra Distance
  + Cosine Distance
  + Hamming Distance
* The Fisher’s exact test is used when you want to conduct a chi-square test, but one or more of your cells has an expected frequency of five or less.  Remember that the chi-square test assumes that each cell has an expected frequency of five or more, but the Fisher’s exact test has no such assumption and can be used regardless of how small the expected frequency is.
* Make your code PEP8 compatible. Pycodestyle. Autopep8. Pylint.
* magic functions: %%prun, %%memit
* WoE and IV concept: The weight of evidence (WOE) and information value (IV) provide a great framework for for exploratory analysis and variable screening for binary classifiers. WOE and IV have been used extensively in the credit risk world for several decades, and the underlying theory dates back to the 1950s. However, it is still not widely used outside the credit risk world and it is a somewhat underserved area in R.
  + WOE describes the relationship between a predictive variable and a binary target variable.
  + IV measures the strength of that relationship.
  + Excellent blog article: <https://multithreaded.stitchfix.com/blog/2015/08/13/weight-of-evidence>
* The answer is yes, with the xgboostExplainer R package. The prediction of 21.4% is broken down into the impact of each individual feature. More specifically, it breaks down the log-odds of the prediction, which in this case is -1.299.

-1.41 ::: Baseline (Intercept)   
-1.10 ::: Satisfaction Level (prediction is now -2.51)   
+0.98 ::: Last Evaluation (prediction is now -1.53)   
+0.32 ::: Time Spent At Company (prediction is now - 1.21)   
+0.27 ::: Hours Average Monthly (prediction is now -0.94)   
-0.24 ::: Sales (prediction is now -1.18)   
-0.18 ::: Number of Projects (prediction is now -1.36)  
+0.11 ::: Work Accident (prediction is now -1.25)   
-0.07 ::: Salary (prediction is now -1.32)   
+0.02 ::: Promotion Last 5 Years (prediction is now -1.30)

= -1.299 ::: Prediction

* It’s now clear that the high satisfaction score of the employee does indeed pull the predicted log-odds down (by-1.1), but this is more than cancelled out by the impacts of the last evaluation, time at company and average hours variables, all of which pull the log-odds up.
* <https://scikit-learn.org/stable/modules/clustering.html>
* <https://www.kaggle.com/kashnitsky/topic-7-unsupervised-learning-pca-and-clustering>
* <https://www.kaggle.com/tianyiy/95-accuracy-by-unsupervised-learning>
* <https://www.kaggle.com/farhanmd29/unsupervised-learning>
* <https://github.com/diefimov/MTH594_MachineLearning/blob/master/ipython/Lecture10.ipynb>
* To do Tomorrow
  + Feature Elimination and Retraining
  + Comparative analysis of diffeent explainers
    - SHapley Additive exPlanations  (SHAP) - <https://github.com/slundberg/shap>
    - LIME <https://youtu.be/hUnRCxnydCc>. <https://github.com/marcotcr/lime>
    - Logistic
    - Skater - <https://github.com/datascienceinc/Skater>
    - <On different models>
    - Model-independent explanation methods - <http://lkm.fri.uni-lj.si/rmarko/papers/RobnikSikonjaKononenko08-TKDE.pdf>
    - Interactions of subsets of feature values (<http://lkm.fri.uni-lj.si/xaigor/slo/pedagosko/dr-ui/DKE-Strumbelj-Kononenko-Robnik.pdf>)
    - Variable Importance (<https://arxiv.org/pdf/1801.01489.pdf>)
  + Missing Value and Outlier detection and Removal
  + Whitespace removal and serious-preprocessing
  + Encoding experiments and different encoding tryout
  + Kmodes and other categorical data algos exploration
  + Market Basket Analysis x->y
  + Pre-authorization best practices and use-cases
  + Let’s create a voting ensemble of Explainers
  + Flask Dashboard
  + Preauth use cases
  + Catboost exploration
* **Data pre-processing: address missing values, remove useless instances, possibly discretize continuous features, and address other similar issues. This can be combined with feature selection to reduce the features to those relevant for the problem.**
* Why use LIME. SHAP over Correlation, ANOVA, GLM? - <https://christophm.github.io/interpretable-ml-book/the-future-of-interpretability.html>
* Clustering
  + Using Gower’s similarity coefficient (Gower, 1971) and other dissimilarity measures (Gowda and Diday, 1991) the standard hierarchical clustering methods can handle data with numeric and categorical values (Anderberg, 1973; Jain and Dubes, 1988). However, the quadratic computational cost makes them unacceptable for clustering large data sets. On the other hand, the k-means clustering method (MacQueen, 1967; Anderberg, 1973) is efficient for processing large data sets. Therefore, it is best suited for data mining. However, the k-means algorithm only works on numeric data, i.e., the variables are measured on a ratio scale (Jain and Dubes, 1988), because it minimises a cost function by changing the means of clusters. This prohibits it from being used in applications where categorical data are involved. The traditional approach to converting categorical data into numeric values does not necessarily produce meaningful results in the case where categorical domains are not ordered.
  + In this paper we present two new algorithms that use the k-means paradigm to cluster data having categorical values. The k-modes algorithm (Huang, 1997b) extends the k-means paradigm to cluster categorical data by using (1) a simple matching dissimilarity measure for categorical objects (Kaufman and Rousseeuw, 1990), (2) modes instead of means for clusters and (3) a frequency-based method to update modes in the k-means fashion clustering process to minimise the clustering cost function. The k-prototypes algorithm (Huang, 1997a) integrates the k-means and k-modes processes to cluster data with mixed numeric and categorical values. In the k-prototypes algorithm we define a dissimilarity measure that takes into account both numeric and categorical attributes.
  + >> K-mode clustering on categorical data only by binning service\_qty and age. Create a 2x2 mis-classification matrix with clusters and labels.
  + mca is a [Multiple Correspondence Analysis](http://en.wikipedia.org/wiki/Multiple_correspondence_analysis) (MCA) package for python, intended to be used with [pandas](http://pandas.pydata.org/). MCA is a [feature extraction](http://en.wikipedia.org/wiki/Feature_extraction) method; essentially [PCA](http://en.wikipedia.org/wiki/Principal_component_analysis) for [categorical variables](http://en.wikipedia.org/wiki/Categorical_variable). You can use it, for example, to address [multicollinearity](http://en.wikipedia.org/wiki/Multicollinearity) or the [curse of dimensionality](http://en.wikipedia.org/wiki/Curse_of_dimensionality) with big categorical variables.
* np.set\_printoptions(formatter={'float': '**{: 0.4f}**'.format})
* pd.set\_option('display.precision', 5)
* pd.set\_option('display.max\_columns', 25)
* <https://nbviewer.jupyter.org/github/esafak/mca/blob/master/docs/mca-BurgundiesExample.ipynb>
* Pyclustering - <https://github.com/annoviko/pyclustering/>
  + <https://github.com/annoviko/pyclustering/tree/master/pyclustering>
* Outlier correction in categorical data: <http://pnrsolution.org/Datacenter/Vol3/Issue2/85.pdf>
* Dashboard
  + Dash lets you make rich analytics web apps with only a few hundred lines of Python code. No JavaScript required. Dash is a user interface library for creating analytical web applications. Those who use Python for data analysis, data exploration, visualization, modelling, instrument control, and reporting will find immediate use for Dash.
* Create visual plots
* H2O Python implementation
  + <https://github.com/h2oai/h2o-tutorials/blob/master/h2o-open-tour-2016/chicago/intro-to-h2o.ipynb>
  + AutoML - <https://github.com/h2oai/h2o-tutorials/blob/master/h2o-world-2017/automl/Python/automl_binary_classification_product_backorders.ipynb>

Some applications of unsupervised machine learning techniques include:

1. **Clustering** allows you to automatically split the dataset into groups according to similarity. Often, however, cluster analysis overestimates the similarity between groups and doesn’t treat data points as individuals. For this reason, cluster analysis is a poor choice for applications like customer segmentation and targeting.
2. **Anomaly detection** can automatically discover unusual data points in your dataset. This is useful in pinpointing fraudulent transactions, discovering faulty pieces of hardware, or identifying an outlier caused by a human error during data entry.
3. **Association mining** identifies sets of items that frequently occur together in your dataset. Retailers often use it for basket analysis, because it allows analysts to discover goods often purchased at the same time and develop more effective marketing and merchandising strategies.
4. **Latent variable models** are commonly used for data preprocessing, such as reducing the number of [features](https://www.datarobot.com/wiki/feature/) in a dataset (dimensionality reduction) or decomposing the dataset into multiple components.

* Business understanding
  + Wait times for preauthorized medical care have consequences for patients. 92 percent of the physicians surveyed said that the prior authorization process delays patient access to necessary care; and 78 percent reported that prior authorization can sometimes, often or always lead to patients abandoning a recommended course of treatment.
  + Imagine the efficiencies that could be achieved if the manual workflow of pre-authorizations could be lifted and replaced with automatic approvals that rely on artificial intelligence solutions.