1. All Univariate plots in single cell

sns.countplot(x = "Sex", hue = "Survived", data = train\_dataframe)

plt.show()

sns.countplot(x="Pclass", hue = "Survived", data = train\_dataframe)

plt.show()

sns.countplot(x="SibSp", hue = "Survived", data = train\_dataframe)

plt.show()

sns.countplot(x="Parch", hue = "Survived", data = train\_dataframe)

plt.show()

sns.countplot(x="Embarked", hue = "Survived", data = train\_dataframe)

plt.show()

sns.catplot(x = "Pclass", hue = "Survived", col = "Sex", kind = "count", data = train\_dataframe)

plt.show()

sns.catplot(x = "Parch", hue = "Survived", col = "Sex", kind = "count", data = train\_dataframe)

plt.show()

sns.catplot(x = "SibSp", hue = "Survived", col = "Sex", kind = "count", data = train\_dataframe)

plt.show()

sns.catplot(x = 'Pclass',y = 'Survived', hue='Sex', data=train\_dataframe, kind = "point") *# equivalent to sns.pointplot*

plt.show()

sns.catplot(x = 'Embarked',hue = 'Survived', col='Pclass', data=train\_dataframe, kind = "count")

plt.show()

* Second option

f,ax=plt.subplots(2,2,figsize=(20,15))

sns.countplot('Embarked',data=data,ax=ax[0,0])

ax[0,0].set\_title('No. Of Passengers Boarded')

sns.countplot('Embarked',hue='Sex',data=data,ax=ax[0,1])

ax[0,1].set\_title('Male-Female Split for Embarked')

sns.countplot('Embarked',hue='Survived',data=data,ax=ax[1,0])

ax[1,0].set\_title('Embarked vs Survived')

sns.countplot('Embarked',hue='Pclass',data=data,ax=ax[1,1])

ax[1,1].set\_title('Embarked vs Pclass')

plt.subplots\_adjust(wspace=0.2,hspace=0.5)

plt.show()

1. Binning of patient age

bins = np.linspace(0, age\_max, 20)

train\_dataframe.loc[:,"Age\_bin"] = np.digitize(train\_dataframe.Age, bins)

* Second way

*# Mapping Age*

dataset.loc[ dataset['Age'] <= 16, 'Age'] = 0

dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1

dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <= 48), 'Age'] = 2

dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3

dataset.loc[ dataset['Age'] > 64, 'Age'] = 4 ;

* Third way

data\_train['AgeGroup'] = ["Baby" if (i>=0 **and** i<5) else "Child" if (i>=5 **and** i<12) else "Teenager" if (i>=12 **and** i<18)

else "Student" if(i>=18 **and** i<24) else "Young Adult" if(i>=24 **and** i<35)

else "Adult" if(i>=35 **and** i<60) else "Senior" if(i>=60) else "Unknown"

for i **in** data\_train.Age ]

passanger\_ageGroup = data\_train.AgeGroup

plt.figure(figsize=(10,7))

sns.countplot(passanger\_ageGroup)

plt.title("data\_train Passanger AgeGroup",color = 'blue',fontsize=15)

plt.show()

HCSC use-case mapping: <=26, 26-42, 42-55, 55-63, 63+

1. Logistic regression hypertuning

C = np.logspace(-4,1,10)

tuned\_parameters = [{"solver":["liblinear"],

"C": C,

"penalty":["l2","l1"]},

{"solver":["newton-cg"],

"C": C,

"penalty":["l2"]}]

scores = ["precision"]

for score **in** scores:

print("Tuning hyperparameters for **%s**" % score)

clf = GridSearchCV(LogisticRegression(), tuned\_parameters, cv = 3,

scoring = "**%s**\_macro" % score)

clf.fit(Xtrain, Ytrain)

print("Best parameters found: **{0}**. Score **{1}\n**".format(clf.best\_params\_, clf.best\_score\_))

print("Grid scores on development set: **\n**")

means = clf.cv\_results\_["mean\_test\_score"]

stds = clf.cv\_results\_["std\_test\_score"]

for mean, std, params **in** zip(means, stds, clf.cv\_results\_["params"]):

print("**%0.3f** (+/-**%0.03f**) for **%r**"

% (mean, std \* 2, params))

print("Detailed classification report:")

print()

print("The model is trained on the full development set.")

print("The scores are computed on the full evaluation set.")

print()

print(clf.best\_estimator\_)

1. SVC hypertuning

from sklearn.svm import SVC

parameters = [{"C":[0.01,0.03,0.1,0.3,1.,3.,10],

"kernel": ["linear", "rbf", "sigmoid"],

"gamma": [0.01,0.03,0.1,0.3,1.,3.,10]}]

clf = GridSearchCV(estimator=SVC(),

param\_grid=parameters, scoring = "accuracy", cv = 5)

clf.fit(Xtrain, Ytrain)

print("Best parameters found: **{0}**. Score **{1}\n**".format(clf.best\_params\_, clf.best\_score\_))

Ypred = clf.predict(Xtest)

accuracy\_dict["svm"] = [accuracy\_score(Ytest, Ypred), clf.best\_estimator\_]

print("Accuracy on the test set: **{0}**%".format(accuracy\_score(Ytest, Ypred)\*100))

1. Use describe include=['O'] for better description of categorical variables
2. Test out all 25-30 classifier models including Catboost and LogosticGAM on both overall as well as approval & denial accuracy, with onehot, target+onehot encoding, showing results in below format. Also refer [this kernel [21]](https://www.kaggle.com/ldfreeman3/a-data-science-framework-to-achieve-99-accuracy/notebook) for comparison table code.

models = pd.DataFrame({

'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',

'Random Forest', 'Naive Bayes', 'Perceptron',

'Stochastic Gradient Decent', 'Linear SVC',

'Decision Tree'],

'Score': [acc\_svc, acc\_knn, acc\_log,

acc\_random\_forest, acc\_gaussian, acc\_perceptron,

acc\_sgd, acc\_linear\_svc, acc\_decision\_tree]})

models.sort\_values(by='Score', ascending=False)

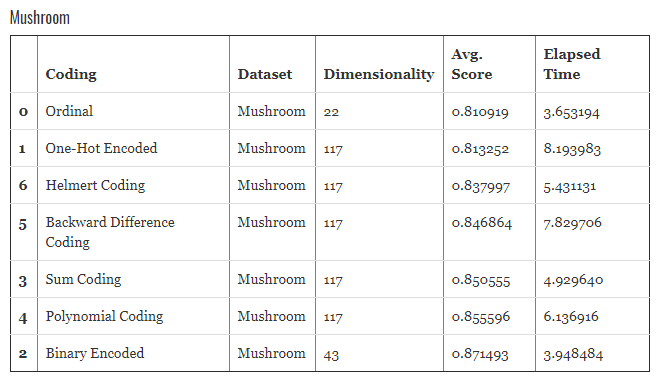
1. Logistic coefficient table view

coeff\_df = pd.DataFrame(train\_df.columns.delete(0))

coeff\_df.columns = ['Feature']

coeff\_df["Correlation"] = pd.Series(logreg.coef\_[0])

coeff\_df.sort\_values(by='Correlation', ascending=False)

1. Different Encoding tryout analysis in this format
2. Learn hyperparameters tuning, Classifier CV comparison and RFE feature selection and voting classifier from [this](https://www.kaggle.com/ldfreeman3/a-data-science-framework-to-achieve-99-accuracy/notebook) kernel. [This](https://www.kaggle.com/yassineghouzam/titanic-top-4-with-ensemble-modeling) kernel is also one of the best.
3. Plot – Age with target variable

*# peaks for survived/not survived passengers by their age*

facet = sns.FacetGrid(titanic\_df, hue="Survived",aspect=4)

facet.map(sns.kdeplot,'Age',shade= True)

facet.set(xlim=(0, titanic\_df['Age'].max()))

facet.add\_legend()

1. Plot – Category with target variable – Factorplot

*# sns.factorplot('Pclass',data=titanic\_df,kind='count',order=[1,2,3])*

sns.factorplot('Pclass','Survived',order=[1,2,3], data=titanic\_df,size=5)

1. Crosstab Analysis – and its factorplot

pd.crosstab([data.Sex,data.Survived],data.Pclass,margins=True).style.background\_gradient(cmap='summer\_r')

sns.factorplot('Pclass','Survived',hue='Sex',data=data)

plt.show()

1. Usefulness of Correlation plot – to identify multicollinearity, selecting features that are highly correlated with target variable, to find relations for unsupervised learning and to extract information from high null value features on the basis of its other correlated feature.
2. Value count in colorful style

data['Age\_band'].value\_counts().to\_frame().style.background\_gradient(cmap='summer')

1. Plot multiple confusion matrix

f,ax=plt.subplots(3,3,figsize=(12,10))

y\_pred = cross\_val\_predict(svm.SVC(kernel='rbf'),X,Y,cv=10)

sns.heatmap(confusion\_matrix(Y,y\_pred),ax=ax[0,0],annot=True,fmt='2.0f')

ax[0,0].set\_title('Matrix for rbf-SVM')

y\_pred = cross\_val\_predict(svm.SVC(kernel='linear'),X,Y,cv=10)

sns.heatmap(confusion\_matrix(Y,y\_pred),ax=ax[0,1],annot=True,fmt='2.0f')

ax[0,1].set\_title('Matrix for Linear-SVM')

y\_pred = cross\_val\_predict(KNeighborsClassifier(n\_neighbors=9),X,Y,cv=10)

sns.heatmap(confusion\_matrix(Y,y\_pred),ax=ax[0,2],annot=True,fmt='2.0f')

ax[0,2].set\_title('Matrix for KNN')

y\_pred = cross\_val\_predict(RandomForestClassifier(n\_estimators=100),X,Y,cv=10)

sns.heatmap(confusion\_matrix(Y,y\_pred),ax=ax[1,0],annot=True,fmt='2.0f')

ax[1,0].set\_title('Matrix for Random-Forests')

y\_pred = cross\_val\_predict(LogisticRegression(),X,Y,cv=10)

sns.heatmap(confusion\_matrix(Y,y\_pred),ax=ax[1,1],annot=True,fmt='2.0f')

ax[1,1].set\_title('Matrix for Logistic Regression')

y\_pred = cross\_val\_predict(DecisionTreeClassifier(),X,Y,cv=10)

sns.heatmap(confusion\_matrix(Y,y\_pred),ax=ax[1,2],annot=True,fmt='2.0f')

ax[1,2].set\_title('Matrix for Decision Tree')

y\_pred = cross\_val\_predict(GaussianNB(),X,Y,cv=10)

sns.heatmap(confusion\_matrix(Y,y\_pred),ax=ax[2,0],annot=True,fmt='2.0f')

ax[2,0].set\_title('Matrix for Naive Bayes')

plt.subplots\_adjust(hspace=0.2,wspace=0.2)

plt.show()

1. Missing value visualization using simple one-line heatmap

fig, ax = plt.subplots(figsize=[15,10])

sns.heatmap(train.isna(), ax=ax, cbar=False, yticklabels=False)

ax.set\_title("NaN in each label for train set");

1. Setting data type while loading the data

train = pd.read\_csv('../input/train.csv', header = 0, dtype={'Age': np.float64})

1. Seaborn aesthetics

sns.despine(offset=10, trim=**True**);

sns.set\_style("whitegrid")

sns.boxplot(data=data, palette="deep")

sns.despine(left=**True**)

1. **Box-cox transformation** - Box-Cox transformations aim to normalize variables. These transformations are an alternative to the typical transformations, such as square root transformations, log transformations, and inverse transformations. The main advantage of Box-Cox transformations is that they optimally normalize the chosen variable. Thus, they avoid the need to randomly try different transformations and automatize the data transformation process.

from scipy.stats import boxcox

X\_train\_transformed = X\_train.copy()

X\_train\_transformed['Fare'] = boxcox(X\_train\_transformed['Fare'] + 1)[0]

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X\_train\_transformed\_scaled = scaler.fit\_transform(X\_train\_transformed)

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2).fit(X\_train\_transformed)

X\_train\_poly = poly.transform(X\_train\_transformed\_scaled)

X\_test\_poly = poly.transform(X\_test\_transformed\_scaled)

1. **Feature selection using chi2** - at some point we need to exclude irrelevant features. Otherwise, we will be penalizing the predictive power of our model. You can find a concise introduction to the feature selection subject in Guyon & Elisseeff (2003). In this work, we will use a univariate statistics approach. This approach selects features based on univariate statistical tests between each feature and the target variable. The intuition is that features that are independent from the target variable, are irrelevant for classification.

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

*## Get score using original model*

logreg = LogisticRegression(C=1)

logreg.fit(X\_train, y\_train)

scores = cross\_val\_score(logreg, X\_train, y\_train, cv=10)

print('CV accuracy (original): **%.3f** +/- **%.3f**' % (np.mean(scores), np.std(scores)))

highest\_score = np.mean(scores)

*## Get score using models with feature selection*

for i **in** range(1, X\_train\_poly.shape[1]+1, 1):

*# Select i features*

select = SelectKBest(score\_func=chi2, k=i)

select.fit(X\_train\_poly, y\_train)

X\_train\_poly\_selected = select.transform(X\_train\_poly)

*# Model with i features selected*

logreg.fit(X\_train\_poly\_selected, y\_train)

scores = cross\_val\_score(logreg, X\_train\_poly\_selected, y\_train, cv=10)

print('CV accuracy (number of features = **%i**): **%.3f** +/- **%.3f**' % (i,

np.mean(scores),

np.std(scores)))

*# Save results if best score*

if np.mean(scores) > highest\_score:

highest\_score = np.mean(scores)

std = np.std(scores)

k\_features\_highest\_score = i

elif np.mean(scores) == highest\_score:

if np.std(scores) < std:

highest\_score = np.mean(scores)

std = np.std(scores)

k\_features\_highest\_score = i

*# Print the number of features*

print('Number of features when highest score: **%i**' % k\_features\_highest\_score)

*# Select features*

select = SelectKBest(score\_func=chi2, k=k\_features\_highest\_score)

select.fit(X\_train\_poly, y\_train)

X\_train\_poly\_selected = select.transform(X\_train\_poly

*# Fit model*

logreg = LogisticRegression(C=1)

logreg.fit(X\_train\_poly\_selected, y\_train)

1. Use t-SNE to plot data in 2d color-coded with target variable to identify if approval and denied cases are seperable or not. Use autoencoders if speeration is not present. Refer to [this](https://www.kaggle.com/shivamb/semi-supervised-classification-using-autoencoders) kernel for more information. 
2. Permutation Importance – works on any model

import eli5

from eli5.sklearn import PermutationImportance

perm = PermutationImportance(my\_model, random\_state=1).fit(val\_X, val\_y)

eli5.show\_weights(perm, feature\_names = val\_X.columns.tolist())

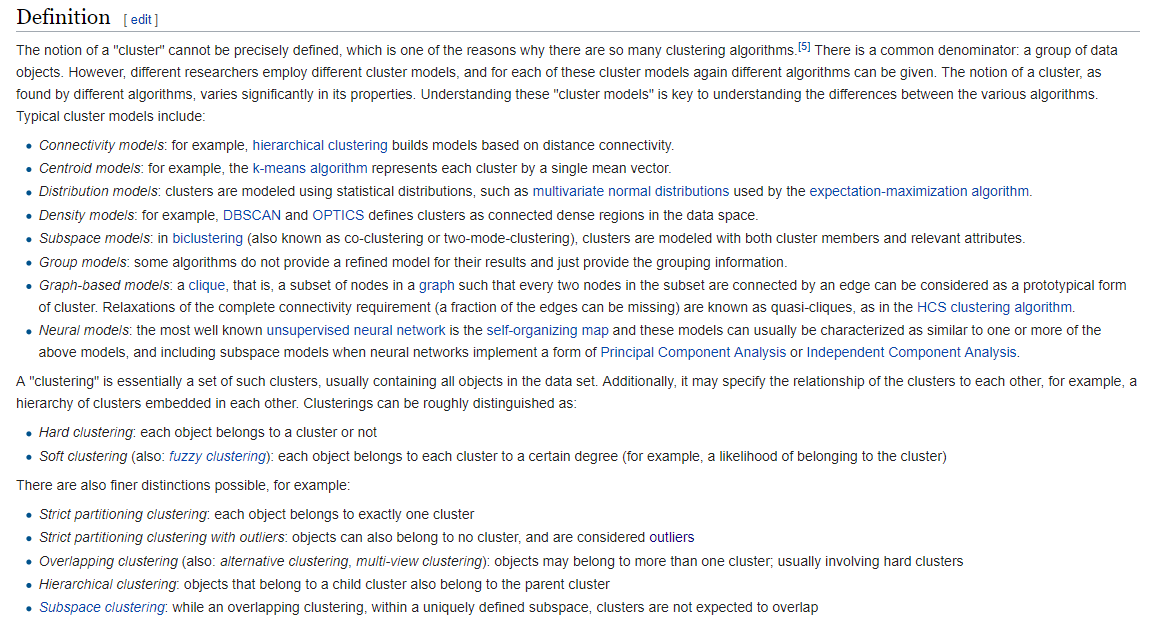
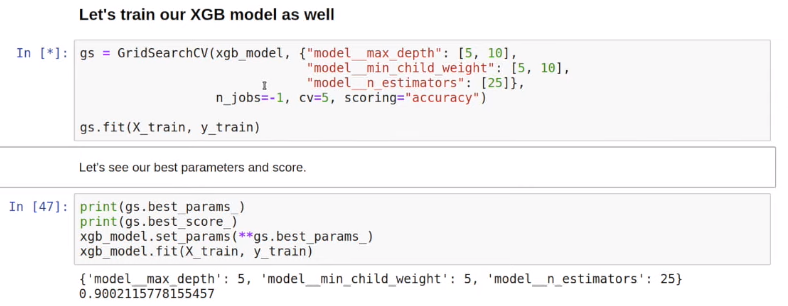
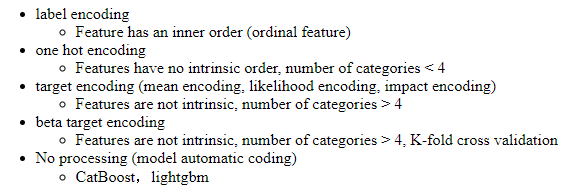
#Explain model globally

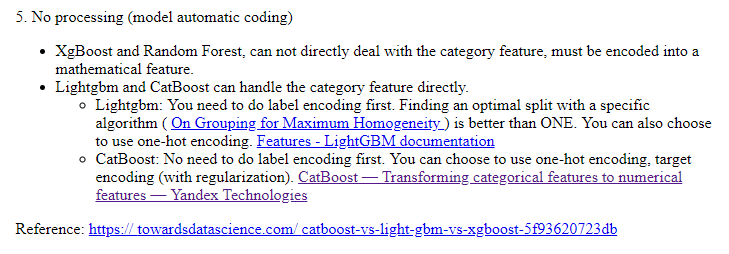
Eli5.show\_weights(model)

#Explain a single explanation

Eli5.show\_prediction(model, observation)

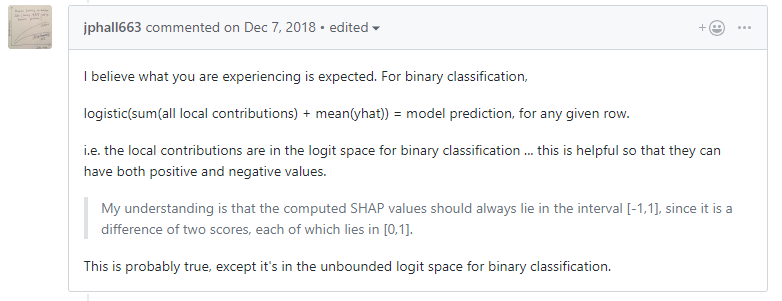
show\_prediction(clf, valid\_xs[1], vec=vec, show\_feature\_values=**True**)

1. [Dealing](https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/) with imbalanced classes
   1. Oversample using SMOTE and MSMOTE and check train, test, approval and denial accuracies
   2. Hypertuned Bagging and Boosting classifiers and check metrics
2. Bagging oob\_score = True gives the idea of test accuracy -> bag\_clf.oob\_score\_. Use warm\_start = 1 during randomizedsearchcv
3. 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.
   1. To train the blender, common approach is hold-out set
   2. There is an open source implementation ‘brew’. Scikit yet not support stacking
4. Unsupervised ML
   1. “In some pattern recognition problems, the training data consists of a set of input vectors x without any corresponding target values. The goal in such unsupervised learning problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine how the data is distributed in the space, known as density estimation.”
   2. Scikit Unsupervised learning [webpage](https://scikit-learn.org/stable/unsupervised_learning.html)
   3. Clustering analysis
      1. To understand clustering – read [this](http://axon.cs.byu.edu/Dan/678/papers/Cluster/Xu.pdf) paper.
      2. Strategy – Take only 2 features and plot scatterplot with decision hue. Then add more features one by one in this scatter plot.
      3. Partitioning-based algorithms: k-Prototypes, k-modes, k-medioids, Squeezer
      4. Hierarchical algorithms: [ROCK](https://www.cis.upenn.edu/~sudipto/mypapers/categorical.pdf), Agglomerative single, average, and complete linkage
      5. Density-based algorithms: HIERDENC, MULIC, CLIQUE
      6. Model-based algorithms: SVM clustering, Self-organizing maps
      7. **Option 1** - 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.
      8. **Why PCA would not work here?** - 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.
      9. **One-hot encoding is also not a solution because of high cardinal features** - Ralambondrainy (1995) presented an approach to using the k-means algorithm to cluster categorical data. Ralambondrainy’s approach is to convert multiple category attributes into binary attributes (using 0 and 1 to represent either a category absent or present) and to treat the binary attributes as numeric in the k-means algorithm. If it is used in data mining, this approach needs to handle a large number of binary attributes because data sets in data mining often have categorical attributes with hundreds or thousands of categories. This will inevitably increase both computational and space costs of the k-means algorithm. The other drawback is that the cluster means, given by real values between 0 and 1, do not indicate the characteristics of the clusters.
      10. **Conceptual clustering algorithms** - Conceptual clustering algorithms developed in machine learning cluster data with categorical values (Michalski and Stepp, 1983; Fisher, 1987; Lebowitz, 1987) and also produce conceptual descriptions of clusters. The latter feature is important to data mining because the conceptual descriptions provide assistance in interpreting clustering results. Unlike statistical clustering methods, these algorithms are based on a search for objects which carry the same or similar concepts. Therefore, their efficiency relies on good search strategies. For problems in data mining, which often involve many concepts and very large object spaces, the concept-based search methods can become a potential handicap for these algorithms to deal with extremely large data sets. Some popular ones include CLARANS (Ng and Han, 1994), DBSCAN (Ester et al., 1996) and BIRCH (Zhang et al., 1996). These algorithms are often revisions of some existing clustering methods. Again, these algorithms still target on numeric data and cannot be used to solve massive categorical data clustering problems.
      11. **K-mode and K-prototype is the solution** - 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. Paper - [Kmodes](https://arxiv.org/ftp/cs/papers/0603/0603120.pdf). Explanation and [Evaluation](http://www.cs.ust.hk/~qyang/Teaching/537/Papers/huang98extensions.pdf).
      12. More advanced methods – [Fuzzy k-modes and Fuzzy k-mode centroids](http://biosoft.kaist.ac.kr/~dhlee/pubs/papers/2004FuzzyClustgeringCategorical.pdf)
      13. Paper - [A k-mean clustering algorithm for mixed numeric](http://edu.cs.uni-magdeburg.de/EC/lehre/sommersemester-2013/wissenschaftliches-schreiben-in-der-informatik/publikationen-fuer-studentische-vortraege/kMeansMixedCatNum.pdf).
      14. Github - <https://github.com/nicodv/kmodes>
   4. Visually identify if there is a clear distinction between approval and denial
   5. Correlation Heatmap
   6. Bivariate and Multivariate visualization
   7. Health insurance fraud analytics
   8. Association rules learning (apriori and eclat)
   9. Bayesian Networks (DAGs)
   10. PCA and TSNE on target encoding data
   11. Dimension reduction of categorical data - Refer [this](https://www.researchgate.net/publication/286927869_Dimensionality_Reduction_of_Categorical_Data_Comparison_of_HCA_and_CATPCA_Approaches) papar.
5. Important features
   1. Procedure and Diagnosis code
   2. Length of stay
   3. Frequency of service
   4. Treatment type
   5. Facility provider type
   6. Patient age
6. **Is Approved/Reject right? Or should we consider Approved and Not-approved (adding all other categories), similar to that of HCSC use-case**
7. 
8. 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.
9. 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
10. Correlation analysis between 2 categorical variables – Chi2 test.
    1. Null hypothesis: they are independent, Alternative hypothesis is that they are correlated in some way.
    2. Correlation between independent variables should be low (no multicollinearity) and correlation between an independent and dependent variable should be high.
    3. For categorical, we use lambda, Cramer V instead of Pearson as a metric.
    4. 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.
11. Different types of distance metrics
    1. Sum of Absolute Distance
    2. Sum of Squared Distance
    3. Mean-Absolute Error
    4. Euclidean Distance
    5. Manhattan Distance
    6. Chessboard Distance
    7. Minkowski Distance
    8. Canberra Distance
    9. Cosine Distance
    10. Hamming Distance
12. Make your code PEP8 compatible. Pycodestyle. Autopep8. Pylint.
13. magic functions: %%prun, %%memit
14. WoE and IV concept: The weight of evidence (WOE) and information value (IV) provide a great framework 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.
    1. WOE describes the relationship between a predictive variable and a binary target variable.
    2. IV measures the strength of that relationship.
    3. Excellent blog article: <https://multithreaded.stitchfix.com/blog/2015/08/13/weight-of-evidence>
15. Comparative analysis of different explainers
    1. Skater - <https://github.com/datascienceinc/Skater>
    2. Model-independent explanation methods - <http://lkm.fri.uni-lj.si/rmarko/papers/RobnikSikonjaKononenko08-TKDE.pdf>
    3. Interactions of subsets of feature values (<http://lkm.fri.uni-lj.si/xaigor/slo/pedagosko/dr-ui/DKE-Strumbelj-Kononenko-Robnik.pdf>)
    4. Variable Importance (<https://arxiv.org/pdf/1801.01489.pdf>)
16. **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.
17. Why use LIME. SHAP over Correlation, ANOVA, GLM? - <https://christophm.github.io/interpretable-ml-book/the-future-of-interpretability.html>
18. Clustering
    1. 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.
    2. 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.
    3. >> K-mode clustering on categorical data only by binning service\_qty and age. Create a 2x2 mis-classification matrix with clusters and labels.
    4. 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.
    5. MCA - <https://nbviewer.jupyter.org/github/esafak/mca/blob/master/docs/mca-BurgundiesExample.ipynb>
    6. Pyclustering - <https://github.com/annoviko/pyclustering/>. <https://github.com/annoviko/pyclustering/tree/master/pyclustering>
19. Outlier correction in categorical data: <http://pnrsolution.org/Datacenter/Vol3/Issue2/85.pdf>
20. 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.
21. 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 in a dataset (dimensionality reduction) or decomposing the dataset into multiple components.
22. *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.*
23. Build a Logistic explainer
24. Try LIME on label/target encoder
25. This is how we do grid search hypertuning. [This](https://github.com/scikit-learn-contrib/categorical-encoding/blob/master/examples/grid_search_example.py) code is also good.
26. Save LIME explanation in html using *exp.save\_to\_file(‘xx.html’)*
27. How to deal with high cardinal feature procedure and diagnosis code
    1. AS IS – One Hot encoding
    2. [Target encoding](https://github.com/scikit-learn-contrib/categorical-encoding/blob/master/category_encoders/target_encoder.py)
       1. TargetEncoder (verbose=1, cols=None, drop\_invariant=False, return\_df=True, handle\_missing='error', handle\_unknown='error', min\_samples\_leaf=5, smoothing=5.0)
       2. means **=** df**.**groupby('x\_0')['y']**.**mean()
       3. df['x\_0'] **=** df['x\_0']**.**map(means)
       4. The trick is to “smooth” the average by including the average rating over all movies. In other words, if there aren’t many ratings we should rely on the global average rating, whereas if there enough ratings then we can safely rely on the local average. Refer [this](https://maxhalford.github.io/blog/target-encoding-done-the-right-way/).[this](https://www.wikiwand.com/en/Additive_smoothing) and [this](https://www.wikiwand.com/en/Bayes_estimator#/Practical_example_of_Bayes_estimators)



* 1. Kaggle winner’s comment on target encoding - *One of my main contributions to the team was Bayesian target encoding. The idea is to use bayesian statistics to encode the categorical variables. The cool thing is that we can encode not only the target mean, but other statistics like the median, mode, variance, skewness, and kurtosis using the same framework. We found that this style of target encoding outperforms the built-in LightGBM categorical encoding.* [*https://www.kaggle.com/mmotoki/avito-target-encoding*](https://www.kaggle.com/mmotoki/avito-target-encoding)*. Also,* [*this is an excellent link*](https://mattmotoki.github.io/beta-target-encoding.html) *to understand beta (bayesian) target encoding.*
  2. [Weight of Evidence encoding](https://github.com/scikit-learn-contrib/categorical-encoding/blob/master/category_encoders/woe.py)

1. Calculate WoE and IV metric for each category
   1. Excluding Age, Service quantity, Procedure and diagnosis code
   2. Include binned age and service quantity
   3. Include 95% ‘othered’ procedure and diagnosis code
   4. Include 100%ile procedure and diagnosis code
2. **Catboost** classifier
   1. Gradient boosting on decision trees
   2. [documentation](https://tech.yandex.com/catboost/doc/dg/concepts/python-reference_catboostclassifier-docpage/#python-reference_catboostclassifier)
   3. After setting a benchmark, it is time to explore your data
      1. Feature importance
      2. Feature interaction
      3. Per object feature importance (SHAP)
      4. Influential documents
      5. New feature evaluation
   4. Overfitting detector
   5. Missing values support
   6. <https://github.com/catboost/tutorials>
   7. https://github.com/catboost/catboost
3. AdaBoost with Decision Tree - <https://youtu.be/LsK-xG1cLYA>
4. In linear regression interpretation, we are assuming monotonicity (i.e. if one variable goes up/down, the output goes only in a particular direction)
5. Explanation techniques
   1. Decision tree surrogate models - <https://youtu.be/Q8rTrmqUQsU?t=1518>
   2. LIME
   3. SHAP – a silver bullet
      1. <https://medium.com/@gabrieltseng/interpreting-complex-models-with-shap-values-1c187db6ec83>.
6. <http://www.nada.kth.se/~ann/exjobb/sara_engardt.pdf>



1. Sensitivity analysis of your model – a important step
2. Meta frame
   1. Name
   2. Suitable name
   3. Description
   4. Values
   5. Cleaning and validation
   6. Encoding
   7. Transformation
   8. Data type
   9. Missing value
   10. Outlier
   11. Hypotheses (how it impacting the decision)
3. Bayesian Optimization
   1. **Bayes\_opt.** [Github](https://github.com/fmfn/BayesianOptimization). Example - [Simple Bayesian Optimization for LightGBM](https://www.kaggle.com/sz8416/simple-bayesian-optimization-for-lightgbm)
   2. Manual implementation function - <https://thuijskens.github.io/2016/12/29/bayesian-optimisation/>
   3. **Hyperopt** package. <https://anaconda.org/conda-forge/hyperopt>. Example [here](https://towardsdatascience.com/automated-machine-learning-hyperparameter-tuning-in-python-dfda59b72f8a). Another example [here](https://towardsdatascience.com/an-introductory-example-of-bayesian-optimization-in-python-with-hyperopt-aae40fff4ff0).
4. Simple logistic regression log odds plot

xs = np.linspace(-4,4,100)

pl.xlabel("Log odds of winning")

pl.ylabel("Probability of winning")

pl.title("How changes in log odds convert to probability of winning")

pl.plot(xs, 1/(1+np.exp(-xs)))

pl.show()

1. Jupyter markdown skills - <https://codeburst.io/jupyter-notebook-tricks-for-data-science-that-enhance-your-efficiency-95f98d3adee4>.