TAREFAS FEITAS:
- Corr <0.94
- Test Chi2
- P-value < 0,005
- Missing values (utilizando a média por atributo- coluna)
- Association Rules (Apriori 0,06 , corrigir no final)
- Clustering (KMEANS, PCA , KMEANS) – Clustering é sem preprocessamento! Está correcto.

TAREFAS A FAZER :

SMOTE – antes de qualquer algoritmo de classificação

```
>>> from imblearn.over_sampling import SMOTE
>>> sm = SMOTE(random_state=42)
>>> X_res, y_res = sm.fit_resample(X, y)
```

K-NN

1 – MINMAX SCALER()
2 – NORMALIZER()
3 – STARDARTSCALER()

- TESTAR PARA CADA UM DOS EXPERTS E CONSENSUS
- Fazer ML-KNN (testar com todos os experts):

`Na consola:` pip3 install scikit-multilearn

```
from skmultilearn.dataset import load_dataset
from skmultilearn.adapt import MLkNN
classifier = MLkNN(k=3)
prediction = classifier.fit(X_train, y_train).predict(X_test)
```

```
import sklearn.metrics as metrics
```
tirar a accuracy

NAÏVE BAYES

Nao temos que fazer nada

DECISION TREES (CART)

Decision trees tend to overfit on data with a large number of features.

Usar os resultados do PCA/Não usar os resultados PCA

- Usar isto para controlar o overfitting

`max_depth`

Mexe nestes parametros

- Use `min_samples_split` or `min_samples_leaf` to ensure that multiple samples inform every decision in the tree, by controlling which splits will be considered. A very small number will usually mean the tree will overfit, whereas a large number will prevent the tree from learning the data. Try `min_samples_leaf=5` as an initial value. If the sample size varies greatly, a float number can be used as percentage in these two parameters. While `min_samples_split` can create arbitrarily small leaves, `min_samples_leaf` guarantees that each leaf has a minimum size, avoiding low-variance, over-fit leaf nodes in regression problems. For classification with few classes, `min_samples_leaf=1` is often the best choice.

RANDOM FORESTS

Usar os resultados do PCA/Não usar os resultados PCA

```
>>> clf = RandomForestClassifier(n_estimators=100, max_depth=2,
...                              random_state=0)
>>> clf.fit(X, y)
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=2, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
>>> print(clf.feature_importances_)
[0.14205973 0.76664038 0.0282433  0.06305659]
```

CROSS VALIDATION

https://scikit-learn.org/stable/modules/cross_validation.html

PODEMOS USAR PARA TODOS OS CLASSIFICADORES!!!!