KNN Regression

from sklearn.neighbors import KNeighborsRegressor

import matplotlib.pyplot as plt

1.Define X and Y
2.Find k-nearest neighbors
3.Find the average price
knn = KNeighborsRegressor(n_neighbors=n)
knn.fit(X,Y)
knn.score(X_test, y_test)

Categorical Variables (Ordinal & Nominal)

import category_encoders as ce

encoder = ce.OrdinalEncoder(mapping=[{'colname': 'name', 'mapping': {'1': 1, '2': 2}}])
encoder = ce.OrdinalEncoder(cols=['colname'])
encoder.fit(X)
X = encoder.transform(X)
Frequency Encoding
encoder = ce.CountEncoder(cols=['colname'])
One-Hot Encoding
encoder = ce.OneHotEncoder()
Target Encoding
encoder = ce.TargetEncoder()

Mean Abolute Error,R2 score,Accuracy score

from sklearn.metrics import mean_absolute_error,r2_score

from sklearn import metrics

from sklearn.metrics import accuracy_score

e = mean_absolute_error(train/test/x/y, predictions)
ep = e*100 / y.mean()
---
r2_score(y_train, preds)
---
validation_e = accuracy_score(y_test, validation_predictions)

Decision Tree

from sklearn.tree import DecisionTreeRegressor

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

1. define X and y
2. regr = DecisionTreeRegressor(random_state=1234,max_depth=int)
3. model = regr.fit(X, y)
4. model.predict(data)
squaredError
squared = (col-col.mean())** 2
squared = sum(squared)/n
Getting the threshold values
regr1.tree_
regr1.tree_.threshold

train_test_split

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=None, train_size=None, random_state=None, shuffle=True)

Methods

from sklearn.model_selection import RandomizedSearchCV

from scipy.stats import randint as sp_randint

from pandas.api.types import is_string_dtype, is_object_dtype,is_numeric_dtype

DataFrame.dropna(axis=0, thresh=int,inplace=False) || lambda x: x.capitalize()
x.to_frame().T #Convert Series to DataFrame.(to_frame)
Df.sort_values(by=colname, axis=int, ascending=True)
.astype(str)
df[colname].fillna(df[colname].median(), inplace=True)

Random Forests

from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators=100, n_jobs=-1, oob_score=True)
rf.fit(X, y)
rf.score(X_train, y_train)
rf.oob_score_
rf.estimators_

Calculating feature importance with rfpimp

from rfpimp import *

I = importances(rf, X_test, y_test)
plot_importances(I, color='#4575b4')

Hyper-parameters

The number of trees, and any other aspect of the model that affects its architecture, statisticians call a hyper-parameter.

Train,Validate,Test

15% test - 15% validation, 70% train

df_dev, df_test = train_test_split(df, test_size=0.15)
df_train, df_valid = train_test_split(df_dev, test_size=0.15)