Kaggle Quick Start

An incomplete Starter Kit for House Price Prediction with Regression

Woodmark

1. Quick Start in Python or R

	Python	R
Import necessary libraries	import numpy as np import pandas as pd import matplotlib.pyplot as plt from scipy.stats import skew from sklearn import linear_model from sklearn.preprocessing import StandardScaler from sklearn.model_selection import cross_val_score	install.packages('moments') install.packages("glmnet") library(moments) library(glmnet)
Load input data	train = pd.read_csv("input/train.csv") test = pd.read_csv("input/test.csv")	getwd() setwd() train <- read.csv("train.csv") test <- read.csv("test.csv")
Check data dimensions	print("Number of features in training set: {}".format(train.shape[1])) print("Number of training data entries: {}".format(train.shape[0])) print("Number of test data entries: {}".format(test.shape[0]))	dim(train) dim(test) str(train) str(test) summary(train) summary(test)
Where is the ID / the training prices?	print("First column in both sets is: {}".format(train.columns[0])) print("Last column in training set is: {}".format(train.columns[-1]))	-
Make index column the actual data frame index	train.set_index('Id', inplace=True) test.set_index('Id', inplace=True)	-
Split off the price column from the training data:	train_price = train["SalePrice"] train.drop("SalePrice", axis=1, inplace=True)	train_price<- train$SalePrice train$SalePrice <- NULL
Check value ranges of features variables:	%matplotlib inline plt.rcParams['figure.figsize'] = (16, 6) train.boxplot(showfliers=False, rot=90) plt.show()	nums <- sapply(train, is.numeric) boxplot(train[,nums], use.cols=TRUE, las = 2)

Deal with numerical features:	# extract locations of numerical features
	num_feat = (train.dtypes != "object").as_matrix()
	
	train_num = train.iloc[:, num_feat]
	test_num = test.iloc[:, num_feat]
		num_feat <- sapply(train, is.numeric)
		num_feat2<- sapply(test, is.numeric)
		
		train_num<-train[,num_feat]
		test_num<-test[,num_feat2]

Check positivity and fill missing values with	print("All numerical values in training set positive? {}" .format(not (train_num < 0).any().any()))	which(train_num<0) which(test_num<0) for(i in 1:ncol(train_num)){ train_num[is.na(train_num[,i]), i] <-

column means:	# fill missing values in training set with column means
	train_num = train_num.fillna(train_num.mean())
	train.iloc[:, num_feat] = train_num
	
	print("All numerical values in test set positive? {}"
	.format(not (test_num < 0).any().any()))
	# fill missing values in test set with TRAINING column means
	test_num = test_num.fillna(train_num.mean())
	test.iloc[:, num_feat] = test_num
		mean(train_num[,i], na.rm = TRUE)}
		
		for(i in 1:ncol(test_num)){
		test_num[is.na(test_num[,i]), i] <- mean(train_num[,i], na.rm = TRUE)}

Check skewness	ax = train_price.plot.density() ax.set_xlim([0,train_price.max()]) ax.set_xlabel("price") ax.set_ylabel("density") plt.show()	skewness(train_num)
	print("Numerical feature columns: {}".format(train.columns[num_feat])) print("Skeweness of numerical training features: {}" .format(skew(train_num)))	-

log(1+p) transform of skewed features	skewed = (np.absolute(skew(train_num)) > 1)
	train_num.iloc[:, skewed] = np.log1p(train_num.iloc[:, skewed])
	test_num.iloc[:, skewed] = np.log1p(test_num.iloc[:, skewed])
	train_price = np.log1p(train_price)
	
	print("Skeweness of numerical training features after transformation: {}"
	.format(skew(train_num)))
		for (col in colnames(train_num)){ if(abs(skewness(train_num[,col]))>1){ train_num[,col]<-log(1+train_num[,col]) }
		
		for (col in colnames(test_num)){ if(abs(skewness(test_num[,col]))>1){ test_num[,col]<-log(1+test_num[,col]) }

Normalize numerical features	scaler = StandardScaler().fit(train_num)
	train_num = scaler.transform(train_num)
	test_num = scaler.transform(test_num
		for (col in colnames(train_num)){ train_num[,col]<-scale(train_num[,col]) }
		
		for (col in colnames(test_num)){ test_num[,col]<-scale(test_num[,col]) }

Apply the transformed values to the orginial sets	train.iloc[:, num_feat] = train_num test.iloc[:, num_feat] = test_num	train[,num_feat]<-train_num test[,num_feat2]<-test_num
Transform categorical to numerical features	train_test = pd.concat([train, test]) train_test = pd.get_dummies(train_test) train_test.head()	…
Split sets again	train = train_test.iloc[:train.shape[0], :] test = train_test.iloc[train.shape[0]:, :]	…

Linear Regression	lin_reg = linear_model.LinearRegression() scores = cross_val_score(lin_reg, train, train_price, cv=5, scoring='neg_mean_squared_error') print("Mean of 5 CV sqrt MSE: {}".format(np.sqrt(-scores.mean())))	…
Improve the Model	…	…

2. Ressources

- Kaggle tutorial links: https://www.kaggle.com/wiki/Tutorials
- Scikit-Learn Python Machine Learning Library: http://scikit-learn.org/
- Tools:
  - R Studio https://www.rstudio.com/
  - Anaconda https://www.continuum.io/downloads

- Statistics:
  - Statistik: Weg zur Datenanalyse (Ludwig Fahrmeir)
  - Theoretical Statistics: Topics for a Core Course (Robert W. Keener)
- Probability Theory:
  - Wahrscheinlichkeitstheorie (Achim Klenke)
  - Understanding Probability (Henk Tijms)
- Machine Learning:
  - An Introduction to Statistical Learning http://www-bcf.usc.edu/~gareth/ISL/
  - The Elements of Statistical Learning http://statweb.stanford.edu/~tibs/ElemStatLearn/
- Online Courses:
  - Coursera Data Science Specialization https://www.coursera.org/specializations/jhu-data-science
  - Coursera Machine Learning https://www.coursera.org/learn/machine-learning
- Cheat Sheets

  R:
  - https://www.rstudio.com/wp-content/uploads/2016/10/r-cheat-sheet-3.pdf
  - https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

- http://nicolascampione.weebly.com/uploads/1/9/4/1/19411255/r_cheat_sheet.pdf

Python:

- https://perso.limsi.fr/pointal/_media/python:cours:mementopython3-english.pdf
- https://ehmatthes.github.io/pcc/cheatsheets/README.html
- https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PythonForDataScience.pdf

Regression:

- http://www.alisonpearce.net/wp-content/uploads/2013/05/Fun-Reg-Cheat-Sheet.pdf
- http://www.stat.ucla.edu/~rosario/classes/091/112-1b/regression