*Московский государственный технический университет им. Н. Э. Баумана*

***Факультет*** *«Информатика и системы управления»*

***Кафедра*** *«Системы обработки информации и управления»*



*Отчёт по лабораторной работе №3*

*по курсу «****Введение в машинное обучение****»*

*Исполнила: Корнеева А.П., ИУ5-41*

*Проверил:**Гапанюк Ю.Е.*

*Москва 2018 г.*

Условие

Необходимо решить задачу предсказания стоимости дома в зависимости от его характеристик.

1. Провести предподготовку данных
2. Разделить данные
3. Обучить модель из sklearn
4. Реализовать линейную регрессию
5. Эксперименты с моделью

Код и результат работы программы

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from scipy import stats

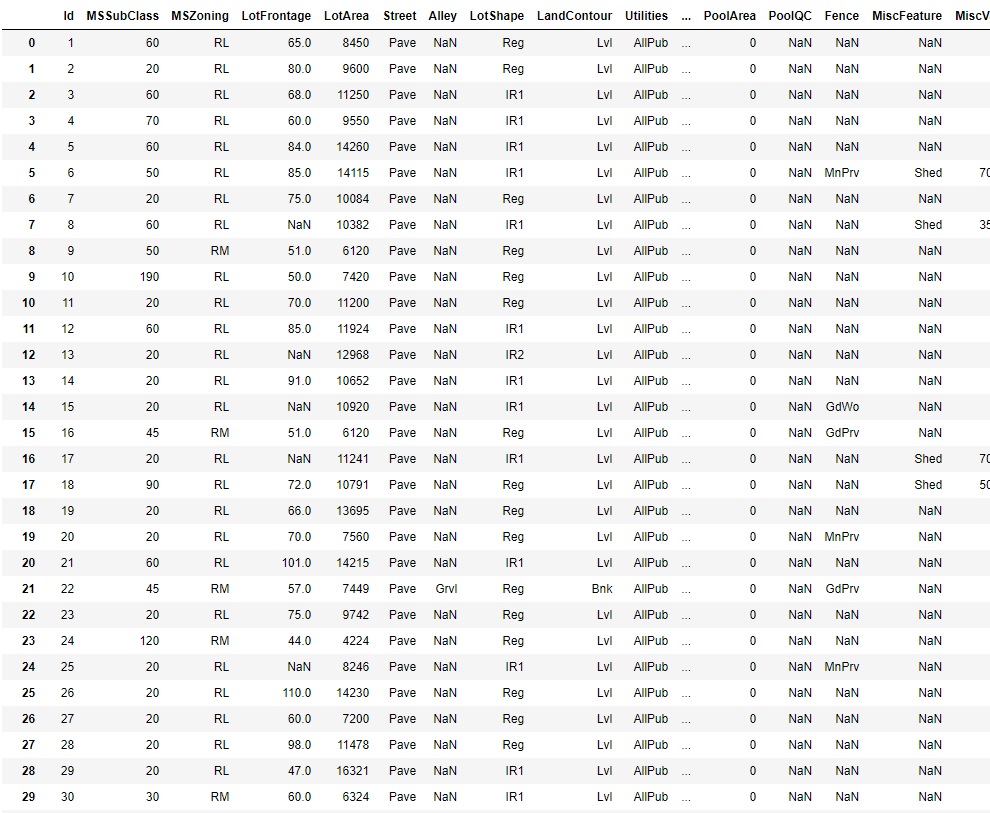
from sklearn.cross\_validation import train\_test\_split

from sklearn import linear\_model

from sklearn.metrics import mean\_absolute\_error, r2\_score

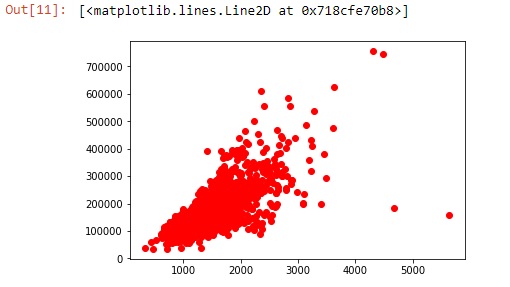
df = pd.read\_csv('./KaggleLab3/train.csv')

df



#распредеение площади домов и цен

plt.plot(df['GrLivArea'], df['SalePrice'], 'ro')



#очистка списка

def cleaning(df):

categorical\_columns = [c for c in df.columns if df[c].dtype.name == 'object']

numerical\_columns = [c for c in df.columns if (df[c].dtype.name != 'object' and c != 'SalePrice')]

answer\_column = [c for c in df.columns if c == 'SalePrice']

#заполняем пустые количественные медианным значением

df = df.fillna(df.median(axis=0), axis=0)

#заполняем пустые категориальные самым частым значением по признаку

df\_describe = df.describe(include=[object]) #получение сводной информации по таблице

for c in categorical\_columns:

df[c] = df[c].fillna(df\_describe[c]['top']) # fillna() - метод для замены отсутствующих значений на числовые

#перевод категориальных фич в числовые

binary\_columns = [c for c in categorical\_columns if df\_describe[c]['unique'] == 2] #бинарные

nonbinary\_columns = [c for c in categorical\_columns if df\_describe[c]['unique'] > 2] #небинарные

for c in binary\_columns:

top = df\_describe[c]['top']

top\_items = df[c] == top

df.loc[top\_items, c] = 0

df.loc[np.logical\_not(top\_items), c] = 1

df\_nonbinary = pd.get\_dummies(df[nonbinary\_columns]) #возврат нового столбца для каждого элемента

#нормализация количественных признаков

df\_numerical = df[numerical\_columns]

df\_numerical = (df\_numerical - df\_numerical.mean()) / df\_numerical.std()

df\_answer = df[answer\_column] #не требуется нормализация

#соединяем всё в таблицу

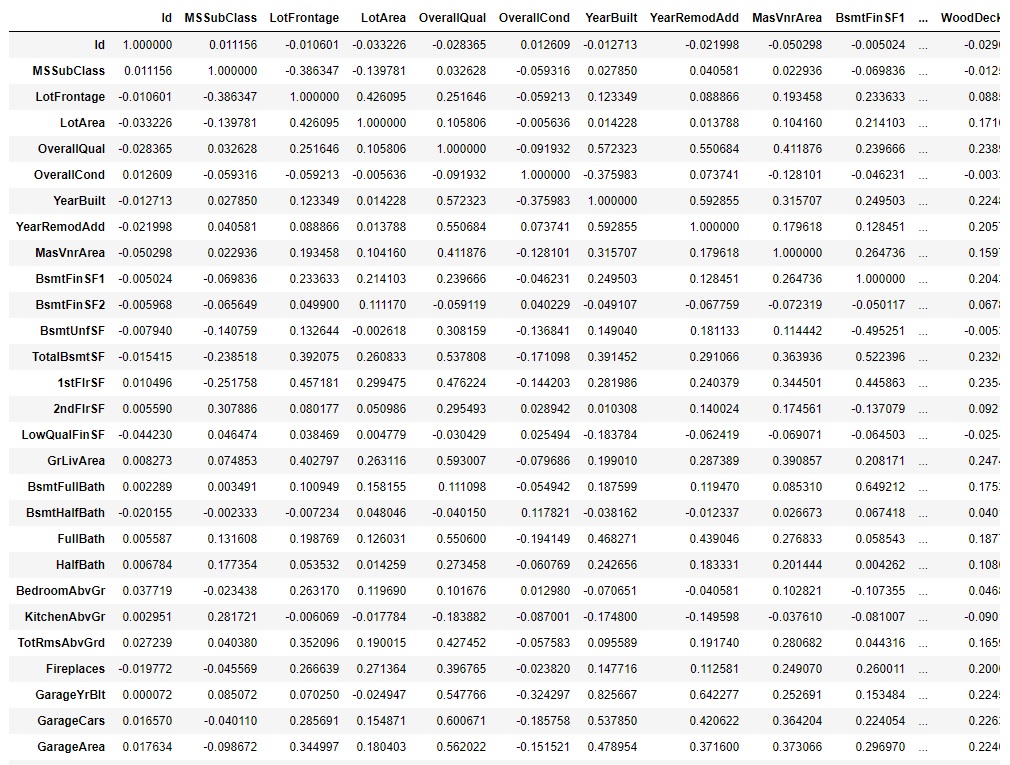
df = pd.concat((df\_numerical, df[binary\_columns], df\_nonbinary, df\_answer), axis=1)

df = pd.DataFrame(df, dtype=float)

return df

df = cleaning(df)

df.corr()

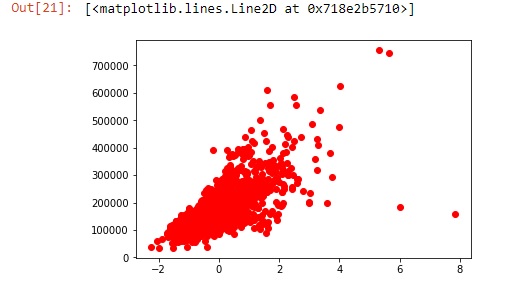


df.corr()['SalePrice'].abs().sort\_values(ascending=False)



df = df[['OverallQual','GrLivArea','GarageCars','TotalBsmtSF','ExterQual\_TA','FullBath','BsmtQual\_Ex','TotRmsAbvGrd','YearBuilt','KitchenQual\_TA','GarageFinish\_Unf','KitchenQual\_Ex','SalePrice']]

plt.plot(df['GrLivArea'], df['SalePrice'], 'ro')



# Разделить данные

x = df.drop(('SalePrice'), axis=1) # входные фичи

y = df['SalePrice'] # ответ

x\_train, x\_valid, y\_train, y\_valid = train\_test\_split(x, y, test\_size = 0.25, random\_state = 11)

#Обучить модель из sklearn, реализовать линейную регрессию

regr = linear\_model.LinearRegression(fit\_intercept=True)

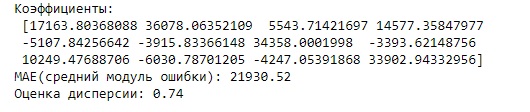
regr.fit(x\_train, y\_train)

y\_valid\_predict = regr.predict(x\_valid)

print('Коэффициенты: \n', regr.coef\_)

print("MAE(средний модуль ошибки): %.2f" % mean\_absolute\_error(y\_valid, y\_valid\_predict))

print('Оценка дисперсии: %.2f' % r2\_score(y\_valid, y\_valid\_predict))



#прогон модели по тестовой выборке

x\_test = pd.read\_csv('./KaggleLab3/test.csv')

y\_test = pd.read\_csv('./KaggleLab3/sample\_submission.csv')

x\_test = cleaning(x\_test)

x\_test = x\_test[['OverallQual','GrLivArea','GarageCars','TotalBsmtSF','ExterQual\_TA','FullBath','BsmtQual\_Ex','TotRmsAbvGrd','YearBuilt','KitchenQual\_TA','GarageFinish\_Unf','KitchenQual\_Ex']]

y\_test = y\_test[['SalePrice']]

#предсказание

y\_test\_predict = regr.predict(x\_test)

print('Коэффициенты: \n', regr.coef\_)

print("MAE: %.2f" % mean\_absolute\_error(y\_test, y\_test\_predict))

print('Оценка дисперсии: %.2f' % r2\_score(y\_test, y\_test\_predict))

