**Московский Государственный Технический Университет имени Н. Э. Баумана**



**Машинное обучение**

**Лабораторная работа №3**

**«Линейная регрессия»**

ИСПОЛНИТЕЛЬ:

ФИО Гладова Анастасия

ИУ5-41 (ИУ5Ц-61Б)

ПРЕПОДАВАТЕЛЬ:

ФИО Гапанюк Ю.Е.

**Москва 2018**

# **Задание:**

Необходимо решить задачу предсказания стоимости дома в зависимости от его характеристик. Задача решается в рамках платформы онлайн-конкурсов по машинному обучению Kaggle. [Ссылка на задание](https://www.kaggle.com/c/house-prices-advanced-regression-techniques).

## **1. Провести предподготовку данных**

(Обязательно) Необходимо перевести категориальные фичи в числовые, отмасштабировать показатели для лучшей обучаемости модели при необходимости (можно провести эксперименты, как это будет влиять на результаты модели). Построить графики по распределнию площадей домой и распределнию цен. Для реализации этой части использовать библиотеки pandas и matplotlib и seaborn.

Необходимо оценить предоставляемые данные, на свое усмотрение предположить несколько возможных зависимостей между признаками и предсказываемыми значениями, проверить гипотезы, построив необходимые графики.

По возможности можно определить, какие признаки являются незначимыми или их доля мала, и объединить такие признаки с другими.

Создать несколько собственных фич на основе своих эвристик и оценить, влияют ли они на качество модели.

Результатом выполнения этого пункта является блок ячеек или скрипт предобработки данных

## **2. Разделить данные**

В этом пункте необходимо поделить данные на обучающую и валидационную выборку. Для этого можно использовать [train\_test\_split](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html). Делить можно в соотношениях 70-90 / 30-10 % соответственно.

## **3. Обучить модель из sklearn**

Следующим шагом необходимо обучить модель линейной регрессии. Для этого нужно использовать класс [LinearRegression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html) из sklearn.

Получить предсказания модели на валидационной части выборки. Оценить результат по метрике [Mean Absolute Error](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error)(MAE) и по метрике, используемой для оценки результатов этого конкурса на [kaggle](https://www.kaggle.com/wiki/RootMeanSquaredLogarithmicError).

## **4. Реализовать линейную регрессию**

На этом шаге необходимо реализовать модель линейной регрессии, используя python самостоятельно. Для этого изначально можно попробовать написать алгоритм для одного обучаемого параметра, а затем написать реализацию общего случая, используя сначала циклы, а затем векторные вычисления из библиотеку numpy. Если есть полное понимание, как нужно реализовать алгоритм для множества обучаемых параметров с использованием векторизации вычислений - можно сразу делать так, главное на защите уметь объяснить.

## **5. Эксперименты с моделью**

На этом шаге вы уже имеете базовую модель, которая делает предсказания. Необходимо прогнать модель на тестовой выборке и отправить решение на kaggle. После этого можно улучшать свой результат, экспериментируя с подготовкой данных и параметрами модели. Рекомендуется смотреть т.н. kernel'ы на kaggle - раздел, где участники соревнований выкладывают код со своими идеями и реализациями. Это может быть очень полезно, как для обучения, так и для реализации новых идей.

*Базовый пример реализации приведен в ноутбуке example.*

## **Заметки**

* При прогоне модели из example, где нет нормировок и используются всего 3 фичи, был получен результат в mae ~ 57к и rmsle ~ 0.409.
* При использовании почти всех фич из выборки mae сократилось до 24к, а rmsle до 0.29
* При замене NaN в фичах можно подставлять наиболее частое значение по фиче из выборки
* Использование всех фич без разбора иногда ухудшает результат
* При переводе качественных фич в цифровые значения и нормализации непрерывных величин удобно написать небольшую функцию и применить ее для всех фич.

# **Листинг:**

## **# In[1]:**

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('C:/Anaconda/train.csv')

df

## **# In[2]:**

#распределение площади домов и цен

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

## **# In[3]:**

#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()

## **# In[4]:**

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

## **# In[5]:**

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')

## **# In[6]:**

# 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))

## **# In[7]:**

#прогон модели по тестовой выборке

x\_test = pd.read\_csv('C:/Anaconda/test.csv')

y\_test = pd.read\_csv('C:/Anaconda/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)

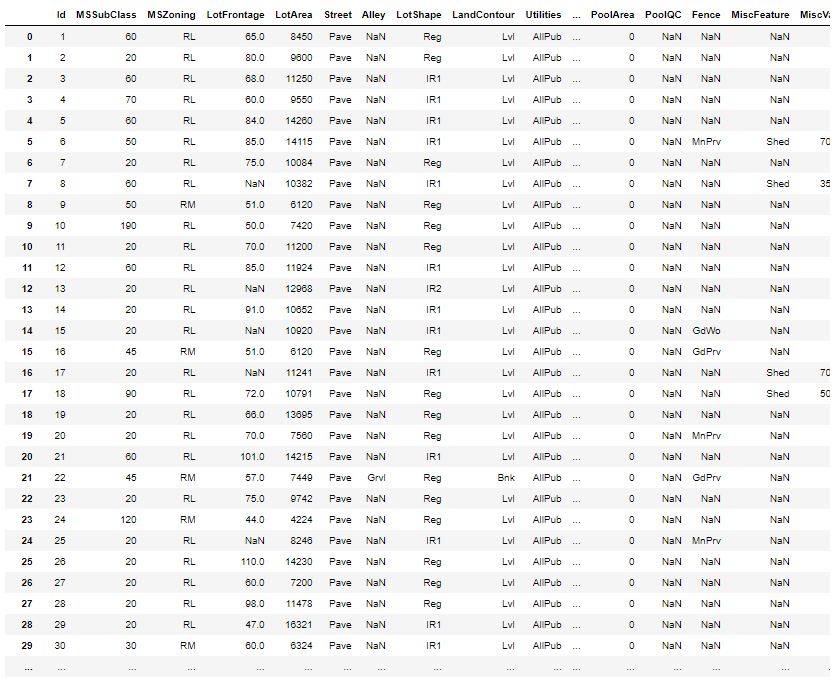
print('Коэффициенты: \n', regr.coef\_)

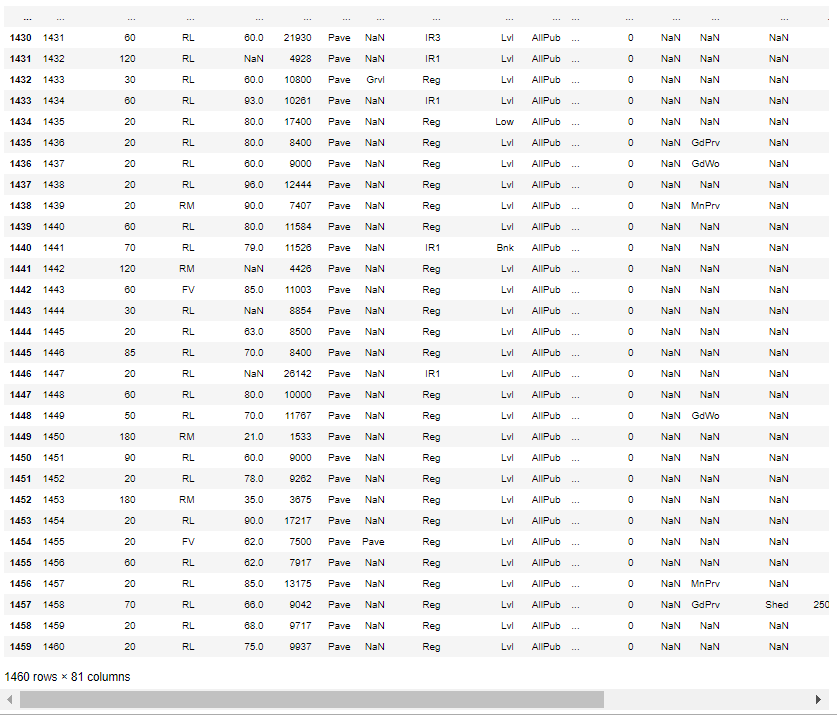
print("MAE: %.2f" % mean\_absolute\_error(y\_test, y\_test\_predict))

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

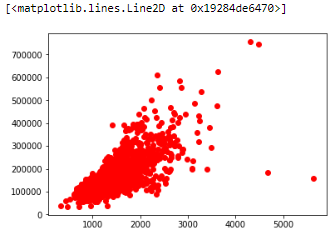
# **Результаты выполнения:**

## **# In[1]:**

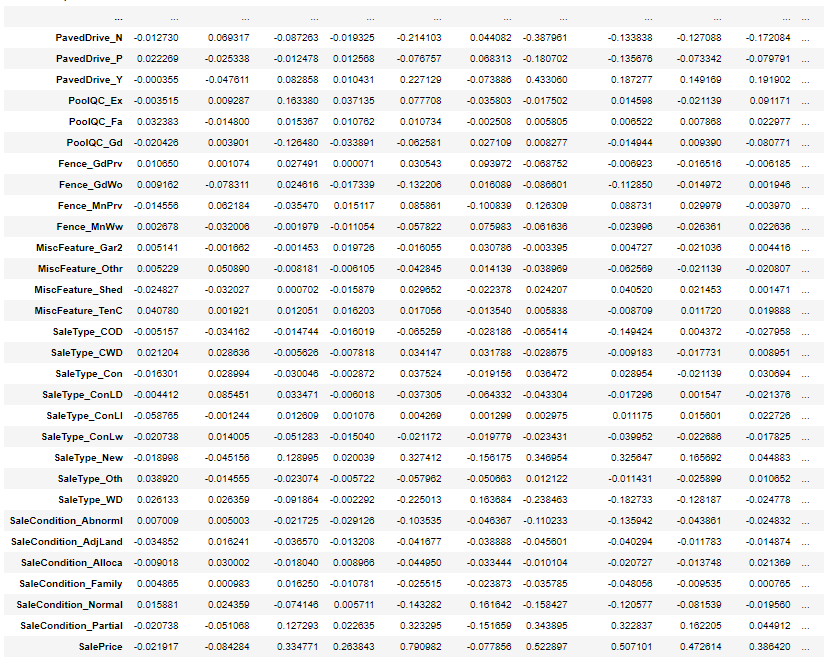
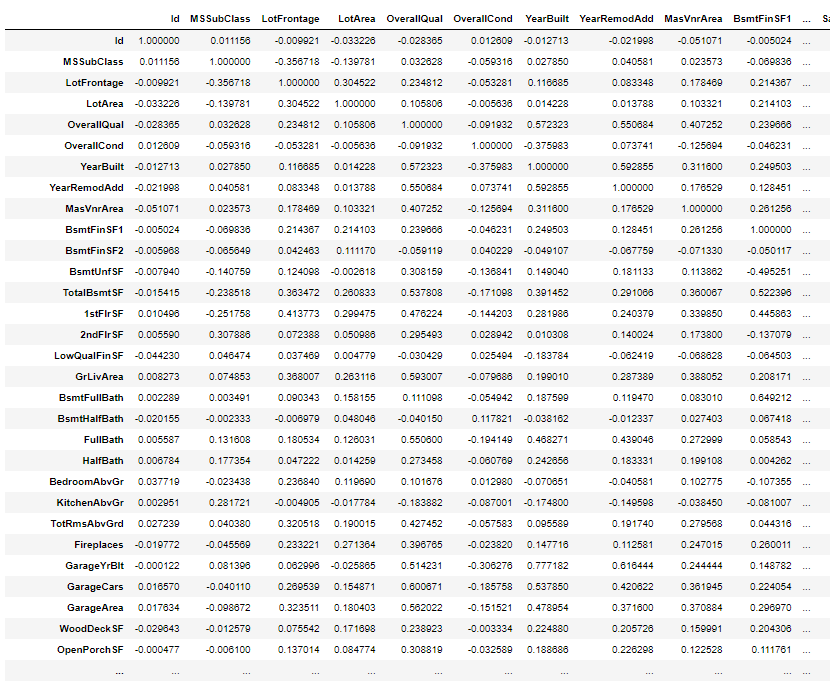




## **# In[2]:**



## **# In[3]:**



## **# In[4]:**

SalePrice 1.000000

OverallQual 0.790982

GrLivArea 0.708624

GarageCars 0.640409

GarageArea 0.623431

TotalBsmtSF 0.613581

1stFlrSF 0.605852

ExterQual\_TA 0.589044

FullBath 0.560664

BsmtQual\_Ex 0.553105

TotRmsAbvGrd 0.533723

YearBuilt 0.522897

KitchenQual\_TA 0.519298

GarageFinish\_Unf 0.513906

YearRemodAdd 0.507101

KitchenQual\_Ex 0.504094

BsmtQual\_TA 0.498545

Foundation\_PConc 0.497734

MasVnrArea 0.472614

Fireplaces 0.466929

GarageYrBlt 0.466754

ExterQual\_Gd 0.452466

ExterQual\_Ex 0.451164

BsmtFinType1\_GLQ 0.434597

HeatingQC\_Ex 0.434543

GarageFinish\_Fin 0.419678

Neighborhood\_NridgHt 0.402149

BsmtFinSF1 0.386420

MasVnrType\_None 0.367456

SaleType\_New 0.357509

...

Functional\_Sev 0.017116

BsmtHalfBath 0.016844

Exterior2nd\_Stone 0.016754

RoofStyle\_Flat 0.016433

PoolQC\_Fa 0.016127

Functional\_Mod 0.016073

SaleCondition\_Alloca 0.015525

Neighborhood\_SawyerW 0.014560

Condition2\_RRAn 0.014510

RoofMatl\_Roll 0.014479

SaleType\_ConLI 0.014371

Utilities 0.014314

Foundation\_Stone 0.012103

ExterCond\_Ex 0.011663

BsmtFinSF2 0.011378

RoofMatl\_ClyTile 0.006897

LotConfig\_FR2 0.006859

Condition1\_RRAn 0.005893

Fence\_GdPrv 0.005152

RoofMatl\_Tar&Grv 0.004921

MiscFeature\_Gar2 0.004744

Condition1\_RRNe 0.004584

LotConfig\_Corner 0.004145

BldgType\_TwnhsE 0.003804

Condition2\_RRAe 0.002993

Foundation\_Wood 0.002711

GarageCond\_Gd 0.000983

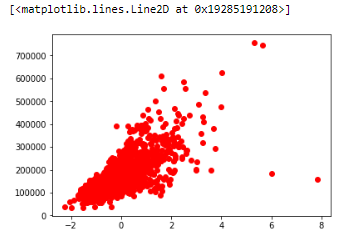
RoofStyle\_Mansard 0.000308

RoofMatl\_Metal 0.000304

BsmtFinType2\_GLQ 0.000076

Name: SalePrice, Length: 286, dtype: float64

## **# In[5]:**



## **# In[6]:**

Коэффициенты:

[20040.67868727 21282.06721516 10568.03838394 8025.47815134

-5591.26260915 -1630.22122407 33991.18852775 2996.11055473

6258.80121185 -9984.24646655 -7607.93499391 29703.13546797]

MAE(средний модуль ошибки): 21247.26

Оценка дисперсии: 0.83

## **# In[7]:**

Коэффициенты:

[20040.67868727 21282.06721516 10568.03838394 8025.47815134

-5591.26260915 -1630.22122407 33991.18852775 2996.11055473

6258.80121185 -9984.24646655 -7607.93499391 29703.13546797]

MAE: 52785.16

Оценка дисперсии: -15.53