Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Самарский национальный исследовательский университет имени академика С.П. Королева»

Лабораторная работа №1

Линейная регрессия
(Linear Regression)

Методические указания
к лабораторным работам по курсу «Большие данные»

Самара 2021

Составитель В.В. Жидченко

УДК 681.3.016

Линейная регрессия (Linear Regression):
Метод. указания к лабораторным занятиям / Самар. ун-т; Сост. В.В. Жидченко.
Самара, 2021. 16 с.

В методических указаниях рассматриваются основы метода построения линейных параметрических моделей, широко применяющегося при анализе “больших данных”. Используются возможности библиотек языка Python: numpy, matplotlib, sklearn, pandas. Методические указания подготовлены на кафедре программных систем.

Работа с языком программирования Python в web-среде. Система Google Colab

В данной работе используются возможности системы Google Colab для создания и запуска программ на языке Python. Указанная система предоставляет вычислительные ресурсы и интерактивный web-интерфейс, работающий по технологии Jupyter Notebook. Система предоставляет виртуальные машины с предустановленными популярными библиотеками Python, поэтому она позволяет познакомиться с различными методами обработки данных без установки программного обеспечения на личный компьютер. Для работы с системой необходимо авторизоваться с помощью аккаунта Google.

Задание 1. Знакомство с системой Colab и реализацией линейной регрессии в библиотеках Python

1) В окне браузера откройте систему Google Golab по ссылке: https://colab.research.google.com/

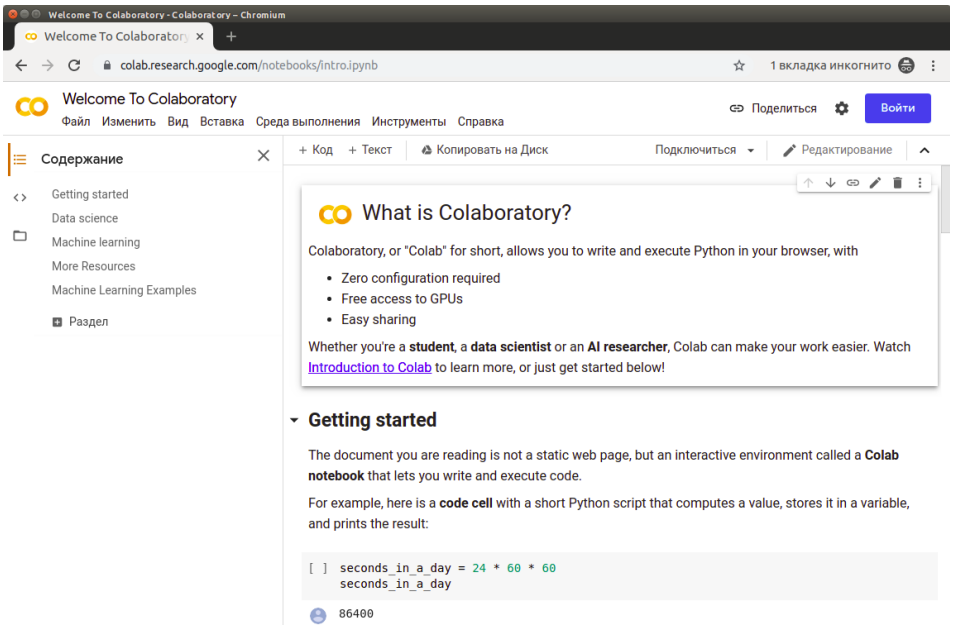


Рис. 1 Главное окно системы Google Colab

2) Выберите меню «Файл» → «Создать блокнот» и авторизуйтесь с аккаунтом Google. Откроется интерфейс редактирования нового блокнота (Notebook). Исходный код на Python вводится в поля ввода, называемые ячейками. Слева от ячейки находится кнопка запуска введенного в ячейку кода на исполнение. Можно создать несколько ячеек и запускать их в произвольном порядке. Переменные, созданные в одной ячейке, видны в других ячейках,

запускаемых после нее. Это позволяет вводить код постепенно, запуская каждый фрагмент и анализируя результат.

3) Загрузите тестовый набор данных, прилагаемый к заданию, на Google Drive под тем же аккаунтом, под которым Вы работаете в Colab.

Этот набор данных содержит значения максимальной температуры воздуха в районе Центрального парка Нью-Йорка для каждого дня в году за период времени с 1874 по 2021 год, предоставляемые Национальной Службой Погоды США (Highest Max Temperature by Day for New York-Central Park Area, NY (ThreadEx)). Данные получены по адресу: https://www.weather.gov/wrh/Climate?wfo=okx, где можно получить аналогичные данные на текущую дату, а также другие погодные статистические данные.

4) Загрузите данные из файла NY_max_temp_daily.csv, введя в ячейку следующий код и запустив его на исполнение, нажав кнопку запуска слева от ячейки или нажав комбинацию клавиш "Ctrl"-"Enter":

```
#Load dependencies
import pandas as pd
import numpy as np
from matplotlib import *
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/drive')

Dset = pd.read_csv('/content/drive/My Drive/NY_max_temp_daily.csv', index_col=0, header=1, na_values=["-"])
```

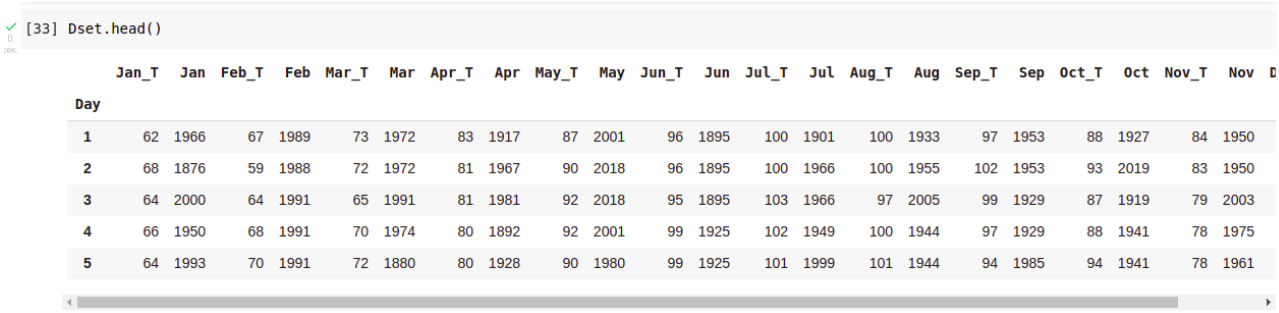
Система Colab запросит Вашего разрешения на доступ блокнота к Вашим файлам на Google Диске или предложит перейти по ссылке в Ваш Google аккаунт и подтвердить доступ к файлу. Во втором случае скопируйте сгенерированный Authorization code и вставьте его в поле «Enter your authorization code:» После этого откроется доступ к файлу из Colab.

В результате выполнения кода функция read_csv() библиотеки Pandas вернет в переменной Dset набор данных (DataFrame), сформированный из содержимого файла. Ознакомьтесь с содержимым DataFrame, вызвав его метод

head(). Для этого создайте новую ячейку, нажав «+ Код» вверху окна. Введите следующий код в новую ячейку:

```
Dset.head()
```

В результате выполнения кода Вы увидите первые несколько строк DataFrame:



[33] Dset.head()

Day	Jan_T	Jan	Feb_T	Feb	Mar_T	Mar	Apr_T	Apr	May_T	May	Jun_T	Jun	Jul_T	Jul	Aug_T	Aug	Sep_T	Sep	Oct_T	Oct	Nov_T	Nov	D
1	62	1966	67	1989	73	1972	83	1917	87	2001	96	1895	100	1901	100	1933	97	1953	88	1927	84	1950	
2	68	1876	59	1988	72	1972	81	1967	90	2018	96	1895	100	1966	100	1955	102	1953	93	2019	83	1950	
3	64	2000	64	1991	65	1991	81	1981	92	2018	95	1895	103	1966	97	2005	99	1929	87	1919	79	2003	
4	66	1950	68	1991	70	1974	80	1892	92	2001	99	1925	102	1949	100	1944	97	1929	88	1941	78	1975	
5	64	1993	70	1991	72	1880	80	1928	90	1980	99	1925	101	1999	101	1944	94	1985	94	1941	78	1961	

Рис. 2 Вывод содержимого DataFrame

DataFrame представляет собой таблицу. Каждая строка этой таблицы содержит значения максимальной температуры воздуха, измеренные в соответствующий день месяца. В файле 31 строка, причем последние две строки содержат пустые значения (NaN) для месяцев, в которых меньше 31 дня. Столбцы данного DataFrame делятся на две подгруппы: температура и годы. Например, в столбце «Jan_T» содержится максимальная температура для каждого дня января, а в столбце «Jan» - год, в котором данное значение температуры было зафиксировано.

5) Создайте новую ячейку и введите в нее следующий код, чтобы преобразовать все значения в тип float и конвертировать градусы Фаренгейта в градусы Цельсия:

```
# Conversion of all values to float
for col in Dset.columns:
	Dset[col] = pd.to_numeric(Dset[col], errors='coerce', downcast='float')
# Conversion of Fahrenheit to Celsius
for i in range(0,12):
	col = Dset.iloc[:,2*i].values
	col = (col-32.0)*5.0/9.0
	Dset.iloc[:,2*i] = col
Dset.tail()
```

```
[16] Dset.tail()
```

Day	Jan_T	Jan	Feb_T	Feb	Mar_T	Mar	Apr_T	Apr	May_T	May	Jun_T	Jun	Jul_T	Jul	Aug_T	Aug	S
27	20.555555	1916.0	22.222221	1997.0	28.333334	1998.0	33.333332	1915.0	35.555557	1880.0	38.333332	1966.0	36.666668	1963.0	38.333332	1948.0	32.22
28	18.888889	1916.0	19.444445	1976.0	28.888889	1945.0	32.222221	2009.0	34.444443	1959.0	35.555557	1991.0	36.111111	1999.0	37.777779	1948.0	31.11
29	20.555555	2002.0	20.555555	1880.0	30.000000	1945.0	31.666666	1974.0	36.111111	1969.0	38.333332	1934.0	37.222221	1949.0	37.222221	1953.0	31.11
30	17.777779	2006.0	NaN	NaN	27.777779	1998.0	32.777779	1942.0	36.111111	1987.0	37.222221	1964.0	36.666668	1988.0	36.666668	1973.0	31.66
31	17.222221	1947.0	NaN	NaN	30.000000	1998.0	NaN	NaN	35.555557	1939.0	NaN	NaN	38.888889	1933.0	37.777779	1953.0	

Рис. 3 Вывод содержимого DataFrame после преобразования типов и значений

Метод DataFrame.tail() выводит несколько последних строк DataFrame. Обратите внимание на значения NaN в феврале, апреле и июне.

Свойство iloc объекта DataFrame позволяет адресовать ячейки таблицы по целочисленному индексу. После преобразования с данными можно работать как с числами.

6) Полезно посмотреть распределение данных в столбцах, построив гистограмму для каждого столбца. Библиотека Pandas позволяет сделать это с помощью одной строки кода. Введите следующий код в новую ячейку:

```
Dset.hist(figsize=(12,12),bins=20)
plt.show()
```

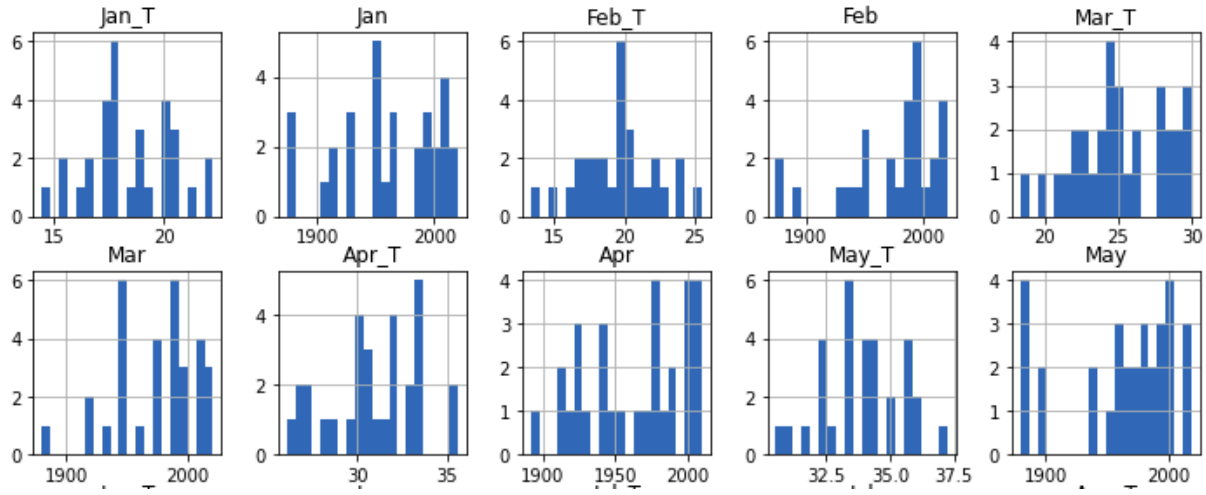


Рис. 4 Гистограммы распределения данных в каждом столбце DataFrame

Каждый столбец в DataFrame обычно интерпретируется как независимая переменная (признак, feature). Построение гистограммы позволяет увидеть распределение значений каждого признака. Исследуемый набор данных имеет довольно специфическую структуру, поэтому на первый взгляд гистограммы не

дают информации об имеющихся зависимостях. Если посмотреть на гистограммы для столбцов с температурой (Jan_T, Feb_T, ...), то можно заметить тенденцию увеличения температуры от января к маю (рассмотрите подписи к осям абсцисс). Построим более наглядное представление данных. Для этого разделим DataFrame на два отдельных набора данных: температура и годы.

7) Введите и запустите следующий код в новой ячейке:

```
# Selecting columns with temperature and years using iloc[row_range,column_range]
Temperatures = Dset.iloc[:,range(0,24,2)]
Years = Dset.iloc[:,range(1,24,2)]
Years.head()
```

8) Отобразите графически значения температуры. Для этого запустите следующий код в новой ячейке:

```
from mpl_toolkits import mplot3d
%matplotlib inline
x = range(1,13)
y = Dset.index
X,Y = np.meshgrid(x,y)
Z = Temperatures
plt.figure(figsize=(10, 10), dpi=80)
ax = plt.axes(projection='3d')
ax.scatter3D(X, Y, Z)
plt.show()
```

В результате построится проекция трехмерного изображения распределения точек из набора данных. По оси x отложены номера месяцев от 1 до 12, по оси y — номера дней месяца, полученные из объекта DataFrame.index, который описывает строки DataFrame. По оси z отложены значения температуры. Каждое значение DataFrame отображается точкой на графике (т.н. Scatter Plot) с помощью метода scatter3d().

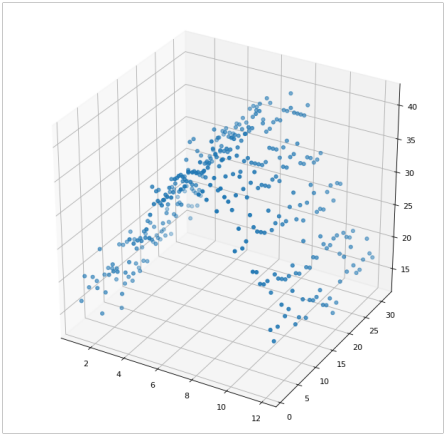


Рис. 5 Scatter Plot значений температуры

На графике видна зависимость температуры от месяца. Температура увеличивается от января к июлю и снова снижается к декабрю. Замеры в каждый из дней месяца дают 28...31 значение для каждого месяца. Изобразим на плоскости зависимость температуры от номера месяца.

```
x = Dset.index
for i in range(0,31):
        plt.scatter(x=range(1,13),y=Temperatures.iloc[i,:])
```

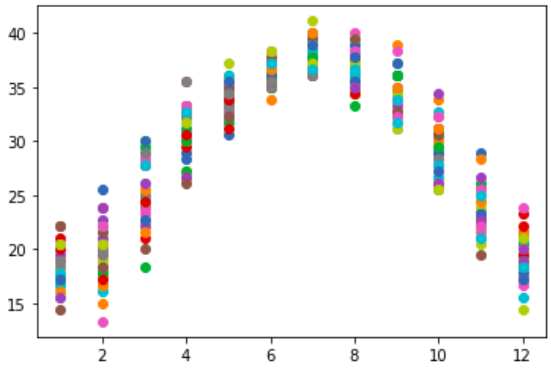


Рис. 6 Scatter Plot значений температуры для каждого месяца

Обратите внимание, что код ячейки выполнился несмотря на то, что для некоторых месяцев значения температуры отсутствуют, т.е. равны NaN. Библиотеки Pandas и Matplotlib распознают значения NaN и учитывают их при обработке данных. Matplotlib не отображает такие значения.

9) Оценим зависимость средней максимальной температуры месяца от номера месяца, используя линейную регрессию. Используем для этого объект LinearRegression библиотеки sklearn.linear_model. Введите и запустите следующий код в новой ячейке:

```
from sklearn.linear_model import LinearRegression
Months = np.array(range(1,13)).reshape(-1,1)
Features = Months
lr = LinearRegression()
lr.fit(X=Features,y=Temperatures.mean())
plt.scatter(x=Months,y=Temperatures.mean())
print("lr.coef_=",lr.coef_[0], "lr.intercept_=",lr.intercept_)
plt.plot(Months,Months*lr.coef_[0]+lr.intercept_)
print("R^2=", lr.score(X=Features,y=Temperatures.mean()))
```

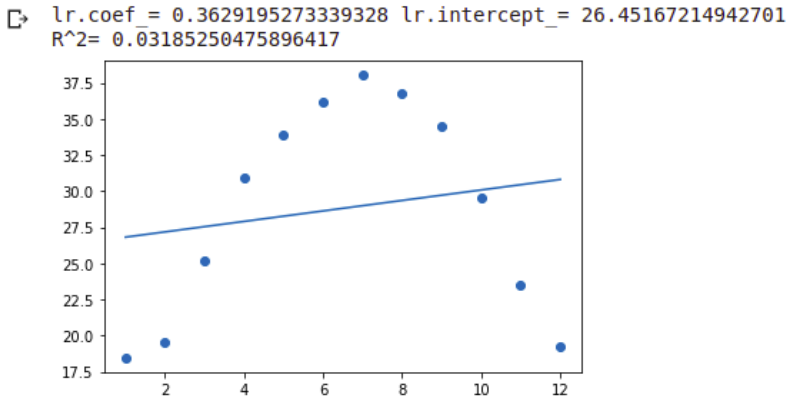


Рис. 7 График линейной модели

Линейная регрессия строится методом LinearRegression.fit(), которая использует матрицу признаков X и вектор значений зависимой переменной y. Для преобразования массива с номерами месяцев в матрицу мы использовали метод numpy.reshape(). После выполнения метода lr.fit() в полях lr.coef_ и lr.intercept_ содержатся значения коэффициентов регрессии и свободного члена соответственно. Эти значения используются для построения графика полученной линейной модели в методе plt.plot().

Метод lr.score() возвращает коэффициент детерминации (статистику R^2) для оценки, полученной с использованием значений признаков и зависимой переменной, переданных в качестве параметров. Для модели, хорошо оценивающей данные, R^2 близок к 1. В приведенном коде коэффициент детерминации вычисляется для значений, использованных при обучении модели. Полученное значение R^2 мало, что показывает нам, что точность линейной модели низкая. Причина этого в том, что исходная зависимость не линейная, а скорее квадратическая.

10) Попробуем улучшить оценку, используя полиномиальную линейную регрессию. Для этого введем новый признак, который будет содержать квадрат номера месяца. Тогда искомая оценка будет иметь вид

$y = B0 + B1*Month + B2*Month*Month$

Модель осталась линейной относительно коэффициентов Bi, т.е. это по-прежнему линейная регрессия, но стала нелинейной относительно признаков.

Введите и запустите следующий код в новой ячейке:

```
MonthsSQ = Months**2
Features = np.hstack((Months,MonthsSQ))
lr = LinearRegression()
lr.fit(X=Features,y=Temperatures.mean())
plt.scatter(x=Months,y=Temperatures.mean())
plt.plot(Months,Months*Months*lr.coef_[1]+Months*lr.coef_[0]+lr.intercept_)
print("R^2=", lr.score(X=Features,y=Temperatures.mean()))
```

```
R^2= 0.9491844183855027
```

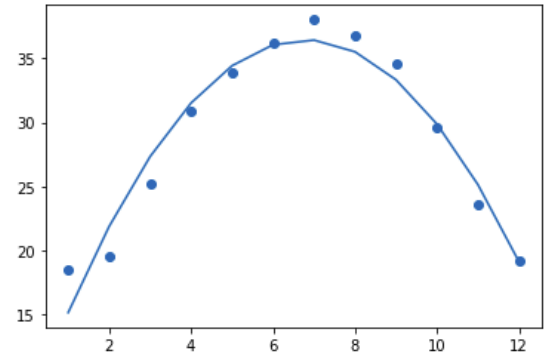


Рис. 8 График модели, полученной с помощью полиномиальной регрессии

Из рисунка видно, что полиномиальная модель гораздо лучше оценивает исходные данные. Коэффициент детерминации существенно увеличился по сравнению с линейной моделью и стал близок к 1.

11) Проверьте самостоятельно, улучшит ли оценку повышение степени полинома до 3. Постройте график полиномиальной модели третьей степени и укажите в отчете коэффициент детерминации, вычисленный для нее.

Задание 2. Анализ набора статистических данных с помощью линейной регрессии

1) В этом задании строится многомерная линейная регрессия для набора статистических данных «California Housing dataset», полученных по результатам переписи населения США 1990 года. С описанием набора можно познакомиться здесь: http://lib.stat.cmu.edu/datasets/houses.zip. Каждая строка набора содержит агрегированные данные в среднем от 1425.5 человек, проживающих компактно на определенной территории штата Калифорния. Площадь области проживания для каждой группы выбрана обратно пропорционально плотности населения. Каждая строка связывает медианную стоимость частного дома с 8 параметрами: медианный доход, медианный возраст дома, среднее количество комнат, среднее количество спальных комнат, население территории, среднее количество жильцов дома, широта и долгота центроида территории, на которой получены данные. Набор данных содержит 20640 строк.

Необходимо построить линейную модель, позволяющую оценить стоимость некоторого дома по значениям перечисленных параметров.

Создайте новый блокнот и загрузите набор данных из репозитория тестовых наборов данных библиотеки Scikit-learn:

```
from sklearn.datasets import fetch_california_housing
cal = fetch_california_housing()
print(cal.DESCR)
```

Объект cal, возвращаемый функцией fetch_california_housing(), содержит набор данных и метаданные, описывающие его. Поле DESCR содержит текстовое описание набора данных. Поле data содержит признаки, поле feature_names — имена признаков, поле target — значения зависимой переменной:

```
print("cal.data.shape:\n",type(cal.data.shape),", value:",cal.data.shape)
print("cal.target.shape:\n",type(cal.target.shape),", value:",cal.target.shape)
print("cal.feature_names:\n",type(cal.feature_names),", value:",cal.feature_names)
```

2) Создайте Pandas DataFrame из набора данных, выполнив следующий код в новой ячейке:

```
import pandas as pd
pd.set_option('precision',4,'max_columns',9)
cal_df = pd.DataFrame(cal.data,columns=cal.feature_names)
cal_df['MedHouseValue'] = pd.Series(cal.target)
cal_df.head()
```

```
cal_df.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseValue
0	8.3252	41.0	6.9841	1.0238	322.0	2.5556	37.88	-122.23	4.526
1	8.3014	21.0	6.2381	0.9719	2401.0	2.1098	37.86	-122.22	3.585
2	7.2574	52.0	8.2881	1.0734	496.0	2.8023	37.85	-122.24	3.521
3	5.6431	52.0	5.8174	1.0731	558.0	2.5479	37.85	-122.25	3.413
4	3.8462	52.0	6.2819	1.0811	565.0	2.1815	37.85	-122.25	3.422

3) Метод DataFrame.describe() вычисляет числовые характеристики для каждого столбца (признака) DataFrame. Он позволяет получить общее представление о значениях каждого признака:

```
[4] cal_df.describe()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseValue
count	20640.0000	20640.0000	20640.0000	20640.0000	20640.0000	20640.0000	20640.0000	20640.0000	20640.0000
mean	3.8707	28.6395	5.4290	1.0967	1425.4767	3.0707	35.6319	-119.5697	2.0686
std	1.8998	12.5856	2.4742	0.4739	1132.4621	10.3860	2.1360	2.0035	1.1540
min	0.4999	1.0000	0.8462	0.3333	3.0000	0.6923	32.5400	-124.3500	0.1500
25%	2.5634	18.0000	4.4407	1.0061	787.0000	2.4297	33.9300	-121.8000	1.1960
50%	3.5348	29.0000	5.2291	1.0488	1166.0000	2.8181	34.2600	-118.4900	1.7970
75%	4.7432	37.0000	6.0524	1.0995	1725.0000	3.2823	37.7100	-118.0100	2.6472
max	15.0001	52.0000	141.9091	34.0667	35682.0000	1243.3333	41.9500	-114.3100	5.0000

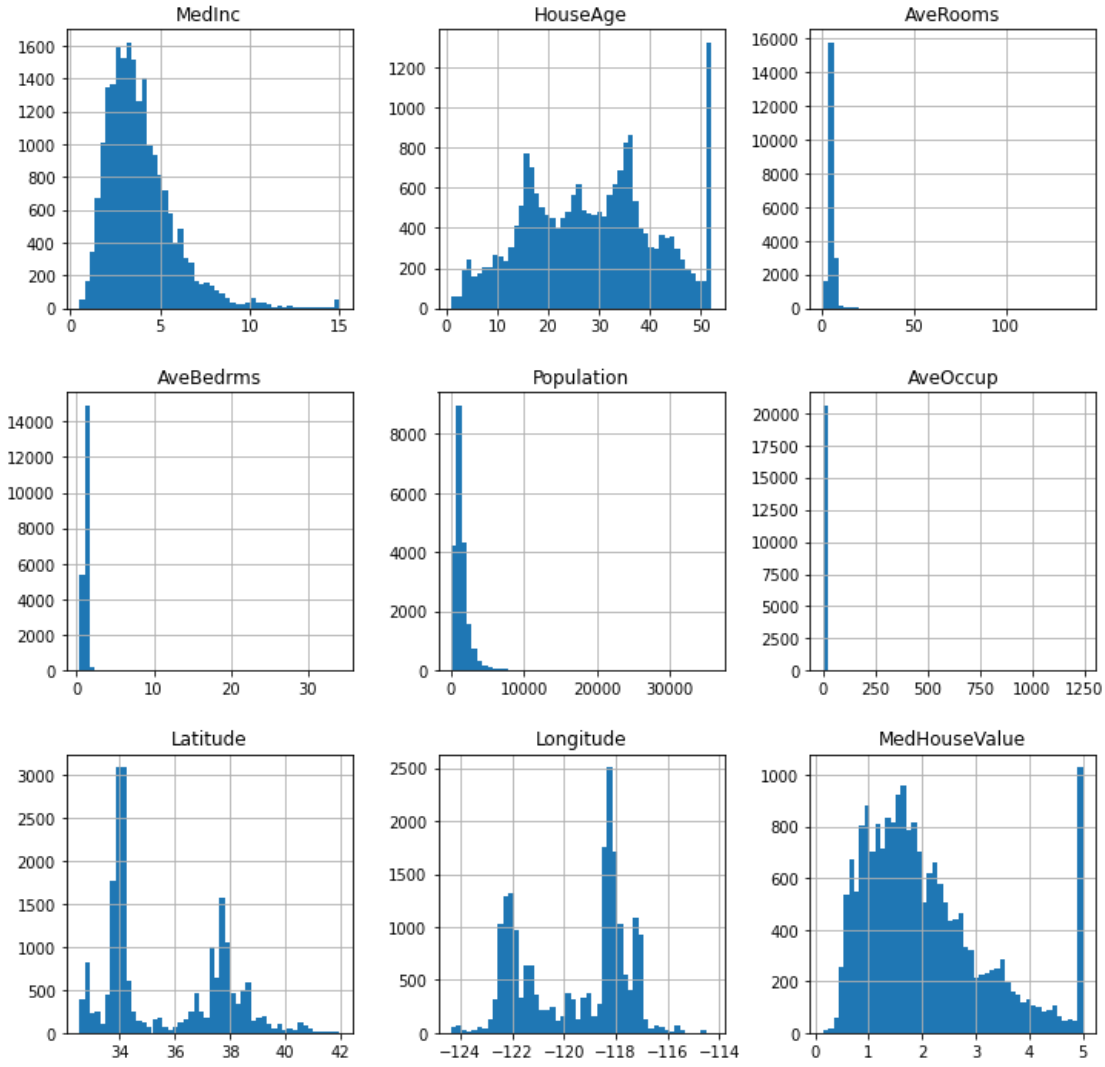
4) Другой полезный метод для ознакомления с набором данных — DataFrame.info():

```
cal_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MedInc         20640 non-null  float64
 1   HouseAge       20640 non-null  float64
 2   AveRooms       20640 non-null  float64
 3   AveBedrms      20640 non-null  float64
 4   Population     20640 non-null  float64
 5   AveOccup       20640 non-null  float64
 6   Latitude       20640 non-null  float64
 7   Longitude      20640 non-null  float64
 8   MedHouseValue  20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

5) Постройте гистограммы для всех столбцов DataFrame:

```
%matplotlib inline
import matplotlib.pyplot as plt
cal_df.hist(figsize=(12,12),bins=50)
plt.show()
```

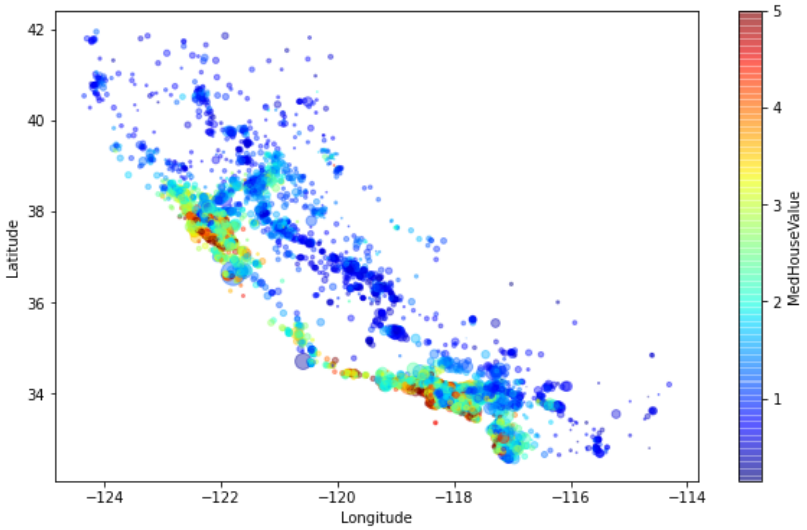


Большое количество элементов с максимальным значением (52 в столбце HouseAge и 5 в столбце MedHouseValue) вызвано тем, что в опросе максимальный возраст дома был ограничен вариантом «52 года или более», а максимальная цена дома — вариантом «500 тыс. долларов или более». Поэтому все дома с возрастом более 52 лет или с ценой более 500 000 имеют одинаковое значение возраста или цены соответственно.

Из гистограмм также видно, что координаты местности по широте и долготе имеют явно выраженные максимумы. Они соответствуют территориям с высокой концентрацией домов.

6) Постройте графическое представление данных в зависимости от координат местности, введя и запустив на выполнение следующий код в новой ячейке:

```
cal_df_sample = cal_df.sample(frac=0.5, random_state=17)
cal_df_sample.plot( kind="scatter", x="Longitude", y="Latitude", alpha=0.4, figsize=(10,6),
                    s=cal_df_sample['Population']/100,
                    c="MedHouseValue", cmap=plt.get_cmap("jet"), sharex=False )
```



Первая строка кода создает подмножество из исходных данных для ускорения процесса визуализации, выбирая половину значений. Вторая строка строит scatter plot, используя координаты местности в качестве значений по осям x и y. Размер каждой точки на графике задается параметром s, который использует население территории для вычисления размера. Цвет точек задается параметром c, который использует цену дома и цветовую шкалу (cmap) для изменения цвета в зависимости от цены дома.

В расположении точек на графике можно заметить очертания побережья Калифорнии.

6.1) (Этот пункт выполнять необязательно.)

Библиотеки Python позволяют нанести график непосредственно на карту. Например, с помощью библиотеки Basemap это можно сделать так:

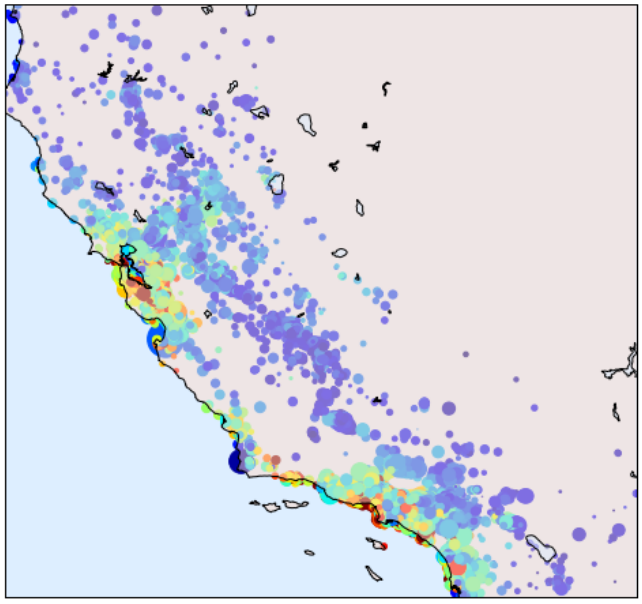
```
# Установка библиотеки Basemap
!apt-get install libgeos-3.5.0
!apt-get install libgeos-dev
!pip install https://github.com/matplotlib/basemap/archive/master.zip
!pip install pyproj==1.9.6

# После установки нужно перезапустить runtime виртуальной среды
```

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
%matplotlib inline

Llcrnrlat = cal_df_sample['Latitude'].min()
Llcrnrlon = cal_df_sample['Longitude'].min()
Urcrnrlat = cal_df_sample['Latitude'].max()
Urcrnrlon = cal_df_sample['Longitude'].max()

fig = plt.figure(figsize=(12, 8))
m = Basemap(projection='cyl', llcrnrlon=Llcrnrlon, llcrnrlat=Llcrnrlat, urcrnrlon=Urcrnrlon,
urcrnrlat=Urcrnrlat, resolution='h')
m.fillcontinents(color="#FFDDCC", lake_color='#DDEEFF', alpha=0.5)
m.drawmapboundary(fill_color="#DDEEFF")
m.drawcoastlines()
Y, X = m(cal_df_sample['Latitude'], cal_df_sample['Longitude'])
plt.scatter(x=X,y=Y,c=cal_df_sample["MedHouseValue"],
            cmap=plt.get_cmap("jet"), s=cal_df_sample["Population"]/40.0 )
plt.show()
```



7) Используя библиотеку Seaborn, постройте зависимости цены дома от каждого из признаков:

```
import seaborn as sns
sns.set_style('whitegrid')
for feature in cal.feature_names:
        plt.figure(figsize=(6,6))
        sns.scatterplot(data=cal_df_sample, x=feature, y='MedHouseValue',
                        hue='MedHouseValue', palette='cool', legend=False)
```

Видна ли на графиках линейная зависимость цены от каких-нибудь признаков? Ответ запишите в отчет.

8) Разделите набор данных на два подмножества: для обучения модели и для ее тестирования:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(cal.data, cal.target, random_state=11)
print("X_train.shape:", X_train.shape)
print("X_test.shape:", X_test.shape)
```

9) Постройте линейную регрессию, используя тренировочный набор данных:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X=X_train,y=y_train)

print('intercept:',lr.intercept_)
for i, name in enumerate(cal.feature_names):
print(f'{name:>10}: {lr.coef_[i]}')
```

10) Оцените и запишите в отчет точность построенной модели, используя тестовый набор данных и статистику R^2.

11) Запишите в отчет функцию регрессии и вычисленную с помощью нее оценку стоимости дома, имеющего следующие параметры:

```
Медианный доход в окружающей местности: 2.4
Возраст дома: 30 лет
Количество комнат: 6
Количество спален: 3
Население в окружающей местности: 1800 человек
Количество жильцов: 5
Широта местности: 37.81
Долгота местности: -117.8
```

12) Оцените стоимость дома с указанными выше параметрами, используя метод LinearRegression.predict(). Ответ запишите в отчет.

Составьте краткий отчет по результатам работы. В отчет необходимо включить Ваши скриншоты с исходным текстом и выводом программы, а также привести ответы на приведенные в задании вопросы.