Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**факультет программной инженерии и компьютерной техники**

**Отчёт по лабораторной работе №3**

по дисциплине

«Системы искусственного интеллекта»

*Выполнил:*

Студент группы P3333

Анисимов Максим Дмитриевич

*Преподаватель:*

Авдюшина Анна Евгеньевна



Санкт-Петербург, 2024

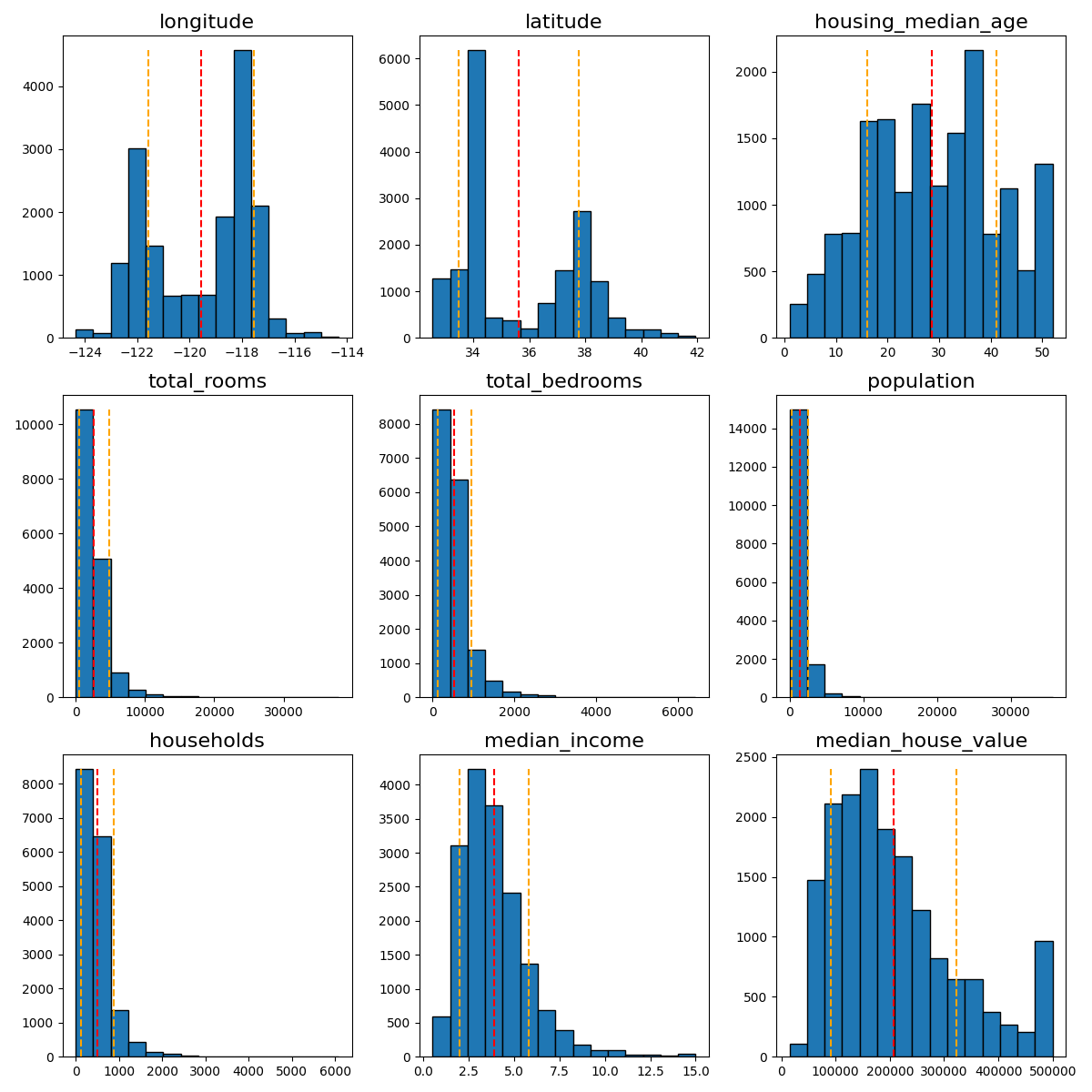
Выбор датасетов:

Студенты с **четным** порядковым номером в группе должны использовать набор данных о жилье в Калифорнии

Получите и визуализируйте (графически) статистику по датасету (включая количество, среднее значение, стандартное отклонение, минимум, максимум и различные квантили).

import pandas as pd  
import numpy as np  
import math  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import StandardScaler, OneHotEncoder  
from sklearn.compose import ColumnTransformer  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import r2\_score  
  
data = pd.read\_csv("california\_housing\_train.csv")  
  
num\_rows = data.shape[0]  
num\_columns = data.shape[1]  
print("Общее количество строк в датасете:", num\_rows)  
print("Общее количество столбцов в датасете:", num\_columns)  
  
X = data.drop(columns=["median\_house\_value"])  
y = data["median\_house\_value"]  
  
rows\_count = X.shape[0]  
cols\_count = X.shape[1]  
  
means = data.mean()  
corrected\_dispersions = np.sum(np.power(data - means, 2)) / (rows\_count - 1)  
standart\_offsets = np.sqrt(corrected\_dispersions)  
mins = data.min()  
maxs = data.max()  
  
bins\_count = 1 + int(math.log(rows\_count, 2))  
  
data\_from = X.join(y)  
titles = data\_from.columns  
fig = plt.figure(figsize=(12, 12))  
  
for i, column in enumerate(titles):  
 ax = fig.add\_subplot(3, 3, i + 1)  
  
 hist = ax.hist(data\_from[column], bins = bins\_count, edgecolor = "black")  
  
 ax.plot([means[i], means[i]], [0, np.max(hist[0])], color = "red", linestyle='--')  
 ax.plot([means[i] - standart\_offsets[i], means[i] - standart\_offsets[i]], [0, np.max(hist[0])], color = "orange", linestyle='--')  
 ax.plot([means[i] + standart\_offsets[i], means[i] + standart\_offsets[i]], [0, np.max(hist[0])], color = "orange", linestyle='--')  
  
 ax.set\_title(titles[i], fontsize=16)  
  
fig.tight\_layout()  
plt.show()

Общее количество строк в датасете: 17000

Общее количество столбцов в датасете: 9

Проведите предварительную обработку данных, включая обработку отсутствующих значений, кодирование категориальных признаков и нормировка.

#вычисление квантилей, средних значений и пр.  
means = data.mean()  
print("Средние значения для каждого столбца:")  
print(means)  
  
std\_deviation = data.std()  
print("\nСтандартное отклонение для каждого столбца:")  
print(std\_deviation)  
  
min\_values = data.min()  
max\_values = data.max()  
print("\nМинимальные значения для каждого столбца:")  
print(min\_values)  
print("\nМаксимальные значения для каждого столбца:")  
print(max\_values)  
  
quantiles = data.quantile([0.25, 0.50, 0.75])  
print("\nКвантили для каждого столбца:")  
print(quantiles)

Вывод:

Средние значения для каждого столбца:

longitude -119.562108

latitude 35.625225

housing\_median\_age 28.589353

total\_rooms 2643.664412

total\_bedrooms 539.410824

population 1429.573941

households 501.221941

median\_income 3.883578

median\_house\_value 207300.912353

dtype: float64

Стандартное отклонение для каждого столбца:

longitude 2.005166

latitude 2.137340

housing\_median\_age 12.586937

total\_rooms 2179.947071

total\_bedrooms 421.499452

population 1147.852959

households 384.520841

median\_income 1.908157

median\_house\_value 115983.764387

dtype: float64

Минимальные значения для каждого столбца:

longitude -124.3500

latitude 32.5400

housing\_median\_age 1.0000

total\_rooms 2.0000

total\_bedrooms 1.0000

population 3.0000

households 1.0000

median\_income 0.4999

median\_house\_value 14999.0000

dtype: float64

Максимальные значения для каждого столбца:

longitude -114.3100

latitude 41.9500

housing\_median\_age 52.0000

total\_rooms 37937.0000

total\_bedrooms 6445.0000

population 35682.0000

households 6082.0000

median\_income 15.0001

median\_house\_value 500001.0000

dtype: float64

Квантили для каждого столбца:

longitude latitude housing\_median\_age total\_rooms total\_bedrooms

0.25 -121.79 33.93 18.0 1462.00 297.00

0.50 -118.49 34.25 29.0 2127.00 434.00

0.75 -118.00 37.72 37.0 3151.25 648.25

population households median\_income median\_house\_value

0.25 790.0 282.00 2.566375 119400.0

0.50 1167.0 409.00 3.544600 180400.0

0.75 1721.0 605.25 4.767000 265000.0

Проведите предварительную обработку данных, включая обработку отсутствующих значений, кодирование категориальных признаков и нормировка.

# Обработка отсутствующих значений  
data = data.dropna()  
  
# Разделение признаков на числовые и категориальные  
data = pd.get\_dummies(data, drop\_first=True)  
  
# Предобработка числовых признаков  
scaler = StandardScaler() # для нормализации (центрирует данные и масштабирует, ср.з. 0; ст. отк. 1)  
data\_scaled = data.copy()  
data\_scaled.iloc[:, :-1] = scaler.fit\_transform(data\_scaled.iloc[:, :-1])

Разделите данные на обучающий и тестовый наборы данных.

X = data\_scaled.drop(columns=['median\_house\_value']) # искл. столбец (ц.п.)  
y = data\_scaled['median\_house\_value'] # цп

Реализуйте линейную регрессию с использованием метода наименьших квадратов без использования сторонних библиотек, кроме NumPy и Pandas (для использования коэффициентов использовать библиотеки тоже нельзя). Использовать минимизацию суммы квадратов разностей между фактическими и предсказанными значениями для нахождения оптимальных коэффициентов.

def linear\_regression(X, y, number\_step=1500, learning\_rate=0.5):  
 number\_samples, number\_functions = X.shape # количество строк и столбцов  
 weights = np.zeros(number\_functions)  
 bias = 0  
  
 for \_ in range(number\_step):  
 # Вычисляются предсказания модели для текущих весов и смещения  
 y\_pred = np.dot(X, weights) + bias  
  
 # Границы  
 dw = (1 / number\_samples) \* np.dot(X.T, (y\_pred - y))  
 db = (1 / number\_samples) \* np.sum(y\_pred - y)  
  
 # Обновляем веса и смещение  
 weights -= learning\_rate \* dw  
 bias -= learning\_rate \* db  
  
 return weights, bias

Постройте три модели с различными наборами признаков.

# Первая модель  
X\_train\_model1 = X\_train  
X\_test\_model1 = X\_test  
  
# Обучение первой модели  
model1 = linear\_regression(X\_train\_model1.values, y\_train.values)  
  
# Предсказание  
y\_prediction\_model1 = np.dot(X\_test\_model1.values, model1[0]) + model1[1] #Для тестовых признаков используются веса и смещение, полученные из обученной модели, для предсказания целевой переменной.  
  
# Оценка производительности на тестовых данных  
r2\_model1 = evaluate\_model(model1, X\_test\_model1.values, y\_test)  
print(f"R^2 для модели 1: {r2\_model1}")  
  
#Вторая модель(исключаем признаки longitude и latitude)  
X\_train\_model2 = X\_train.copy().drop(["longitude", "latitude"], axis=1)  
X\_test\_model2 = X\_test.copy().drop(["longitude", "latitude"], axis=1)  
  
# Обучение второй модели  
weights\_model2, bias\_model2 = linear\_regression(X\_train\_model2.values, y\_train.values) # Веса и смещение модели сохраняются в переменные  
  
# Предсказание  
y\_prediction\_model2 = np.dot(X\_test\_model2.values, weights\_model2) + bias\_model2  
  
# Оценка производительности  
r2\_model2 = r2\_score(y\_test, y\_prediction\_model2)  
print(f"R^2 для модели 2: {r2\_model2}")  
  
  
 # Добавление синтетического признака  
X\_train\_model3 = X\_train.copy()  
X\_test\_model3 = X\_test.copy()  
  
X\_train\_model3["income\_per\_household"] = X\_train\_model3["median\_income"] \* X\_train\_model3["households"] #income\_per\_household, который является произведением признаков median\_income и households  
X\_test\_model3["income\_per\_household"] = X\_test\_model3["median\_income"] \* X\_test\_model3["households"]  
  
# Обучение модели  
weights\_model3, bias\_model3 = linear\_regression(X\_train\_model3.values, y\_train.values)  
  
# Предсказание  
y\_prediction\_model3 = np.dot(X\_test\_model3.values, weights\_model3) + bias\_model3  
  
# Оценка производительности  
r2\_model3 = r2\_score(y\_test, y\_prediction\_model3)  
print(f"R^2 для модели 3: {r2\_model3}")

Вывод:

R^2 для модели 1: 0.6407898493309034

R^2 для модели 2: 0.5615264493805012

R^2 для модели 3: 0.6431176054838503