Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра: «Вычислительная техника»

Дисциплина: «Системы искусственного интеллекта»

Лабораторная работа №3

Линейная регрессия

Вариант 20

Выполнил

Студент группы ИВТАСбд-32

Хасанов И. Н.

Проверил:

преподаватель кафедры «ВТ»

Хайруллин И. Д.

Ульяновск 2024

**Оглавление**

[Постановка задачи 3](#_Toc167462389)

[Реализация 4](#_Toc167462390)

[Вывод 5](#_Toc167462391)

[Приложение. Source.cpp 6](#_Toc167462392)

# Постановка задачи

* Написать программу, которая разделяет исходную выборку на обучающую и тестовую (training set, test set). Использовать стандартные функции (train\_test\_split и др. нельзя).
* С использованием библиотеки scikit-learn обучить модель линейной регрессии по обучающей выборке пример
* Проверить точность модели по тестовой выборке
* Построить модель с использованием полиномиальной функции пример. Построить графики зависимости точности на обучающей и тестовой выборке от степени полиномиальной функции.
* Построить модель с использованием регуляризации пример. На основе экспериментов подобрать параметры для регуляризации. Построить графики зависимости точности модели на обучающей и тестовой выборках от коэффициента регуляризации.

Вариант: SML2010.

# Реализация

Код написан на языке Python. Вначале считываются данные из файла и записываются в датафрейм. После стоится корреляционная матрица через sns.heatmap() для определения параметра для предсказания результатов. Выбирается тот параметр, который имеет наибольшую линейную зависимость с большинством параметров. Затем выбираются переменные, которые имеют наибольшую линейную зависимость с предсказываемым параметром.

Дальше выборка разделяется на обучающую и тестовую и модель линейной регрессии обучается по обучающей модели через linear\_model.LinearRegression(). Точность модели проверяется с помощью r2\_score, то есть находится коэффициент детерминации.

Модель с использованием полиномиальной функции строится с помощью PolynomialFeatures(), находится лучшее значение степени функции и лучшее значение r2\_score, выводятся графики зависимости точности от степени функции.

Для регуляризации на выбор даны “Lasso” и “Ridge”, в данной работе был использован последний: linear\_model.Ridge(). После построения модели были найдены лучшие значения r2\_score и параметра регулялризации, затем были построены графики с зависимостью ошибки (r2\_score) от коэффициента регуляризации для обучающего набора данных и набора тестовых данных.

# Вывод

Проделав данную лабораторную работу, были изучены методы построения моделей линейной регрессии, использования полиномиальной функции и методов регуляризации.

# Приложение. Source.cpp

|  |
| --- |
| import pandas as pd  import numpy as np  from matplotlib import pyplot as plt  import seaborn as sns  from sklearn.model\_selection import train\_test\_split  from sklearn import linear\_model  from sklearn.preprocessing import PolynomialFeatures  from sklearn.metrics import r2\_score  from sklearn.pipeline import Pipeline  from sklearn.linear\_model import SGDRegressor  df = pd.read\_csv('NEW-DATA-1.T15.txt', sep=" ")  df.drop(['#','1:Date'], axis=1, inplace=True)  df = df.iloc[:, 2:]  corrmat = df.corr()  sns.heatmap(corrmat, cmap='magma', square=True)  plt.show()  column\_to\_predict = corrmat.columns[0]  for column in corrmat.columns:  if corrmat[column].sum() > corrmat[column\_to\_predict].abs().sum():  column\_to\_predict = column  print(f'Колонка, значения которой будут предсказываться: {column\_to\_predict}')  min\_correlation = 0.8  features\_condition = (abs(corrmat[column\_to\_predict]) >= min\_correlation) & (corrmat.columns != column\_to\_predict)  features = corrmat.columns[features\_condition]  x\_dataframe = df[features]  print(f'Колонки с лучшей линейной зависимостью: {features.values}')  x\_data = np.array(x\_dataframe)  y\_data = np.array(df[column\_to\_predict])  val\_size = 0.2  x\_train, x\_val, y\_train, y\_val = train\_test\_split(x\_data, y\_data, test\_size=val\_size)  regr = linear\_model.LinearRegression()  regr.fit(x\_data, y\_data)  y\_val\_pred = regr.predict(x\_val)  r2\_val = r2\_score(y\_val, y\_val\_pred)  print(f'Коэффициент детерминации: {r2\_val}')  plt.scatter(x\_val.T[0], y\_val, label="y\_val")  plt.scatter(x\_val.T[0], y\_val\_pred, label="y\_pred")  plt.legend()  max\_degree = 8  r2\_train = [0] \* max\_degree  r2\_val = [0] \* max\_degree  fig, axs = plt.subplots(1, 2)  best\_degree = 1  best\_r2\_score = -1e20  for id, polynomial\_degree in enumerate(range(1, max\_degree + 1)):  polynomial\_features = PolynomialFeatures(degree=polynomial\_degree, include\_bias=False)  linear\_regression = linear\_model.LinearRegression()  pipeline = Pipeline(  [  ("polynomial\_features", polynomial\_features),  ("linear\_regression", linear\_regression),  ]  )  pipeline.fit(x\_train, y\_train)  y\_train\_pred = pipeline.predict(x\_train)  y\_val\_pred = pipeline.predict(x\_val)  r2\_train[id] = r2\_score(y\_train, y\_train\_pred)  r2\_val[id] = r2\_score(y\_val, y\_val\_pred)  if r2\_val[id] >= best\_r2\_score:  best\_r2\_score = r2\_val[id]  best\_degree = polynomial\_degree  print(f'Лучшее значение степени: {best\_degree}. Лучшее значение r2: {best\_r2\_score}')  axs[0].plot(np.linspace(1, max\_degree, max\_degree), r2\_train)  axs[0].set\_title('r2\_train')  axs[1].plot(np.linspace(1, max\_degree, max\_degree), r2\_val)  axs[1].set\_title('r2\_test')  plt.ylim(-30, 5)  fig.tight\_layout()  plt.show()  ridge\_regression\_degree = 8  count\_alpha = 20  ridge\_r2\_train = [0] \* count\_alpha  ridge\_r2\_val = [0] \* count\_alpha  fig, axs = plt.subplots(1, 2)  best\_alpha = 0  best\_r2\_score = -1e20  for id, alpha\_factor in enumerate(np.linspace(0, 1, count\_alpha)):  polynomial\_features = PolynomialFeatures(degree=ridge\_regression\_degree, include\_bias=False)  linear\_regression = linear\_model.Ridge(alpha=alpha\_factor)  pipeline = Pipeline(  [  ("polynomial\_features", polynomial\_features),  ("linear\_regression", linear\_regression),  ]  )  pipeline.fit(x\_train, y\_train)  y\_train\_pred = pipeline.predict(x\_train)  y\_val\_pred = pipeline.predict(x\_val)  ridge\_r2\_train[id] = r2\_score(y\_train, y\_train\_pred)  ridge\_r2\_val[id] = r2\_score(y\_val, y\_val\_pred)  if ridge\_r2\_val[id] >= best\_r2\_score:  best\_r2\_score = ridge\_r2\_val[id]  best\_alpha = alpha\_factor  print(f'Лучшее значение параметра alpha: {best\_alpha}. Лучшее значение r2: {best\_r2\_score}')  axs[0].plot(np.linspace(0, 1, count\_alpha), ridge\_r2\_train)  axs[0].set\_title('ridge\_r2\_train')  axs[1].plot(np.linspace(0, 1, count\_alpha), ridge\_r2\_val)  axs[1].set\_title('ridge\_r2\_val')  plt.ylim(-1, 1)  fig.tight\_layout()  plt.show() |