**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

**ИНСТИТУТ ЦИФРЫ**

**ОТЧЁТ**

**О ВЫПОЛНЕНИИ СЕМЕСТРОВОЙ РАБОТЫ**

Студентки 3 курса, ФИТ-211 группы

**Колесник Полины Олеговны**

Направление 02.03.02 – «Фундаментальная информатика и информационные технологии»

Руководитель:

Подберезин Г. И.

Работа защищена

« »

“ ” 2023 г.

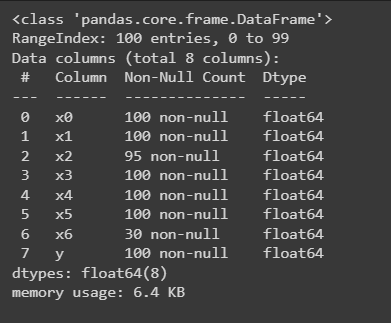
Кемерово 2023 г.

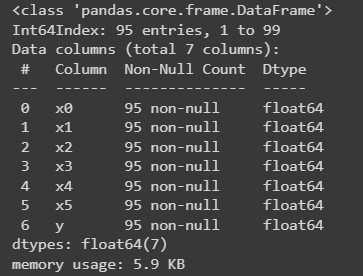
**ОТЧЁТ О ПРОДЕЛАННОЙ РАБОТЕ**

**3 вариант**

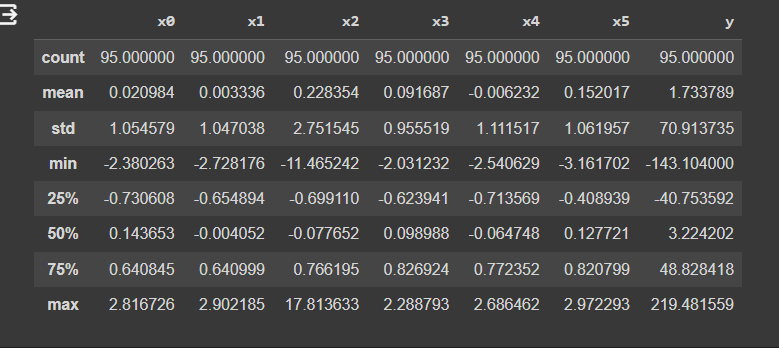
*"""   
Original file is located at  
 https://colab.research.google.com/drive/1pOhD178hR7J58abc0EwchGzW99n9ByQa  
  
# Семестровая работа  
"""*import numpy as np  
import pandas as pd  
  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
from sklearn.model\_selection import train\_test\_split  
  
from sklearn.linear\_model import LinearRegression  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.neighbors import KNeighborsRegressor  
from sklearn.svm import SVR  
# -------------------  
  
from sklearn.model\_selection import GridSearchCV  
  
# ---------------------  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import classification\_report  
  
"""## №1 Задача регрессии (20 баллов)  
  
### 1. Проведите предварительный анализ и подготовку данных, обработайте пропущенные значения.  
"""  
  
df = pd.read\_csv("dataset\_var3.csv", sep=';', names=["x0","x1","x2","x3","x4","x5","x6","y"])  
df.head()

  
  
df.info()

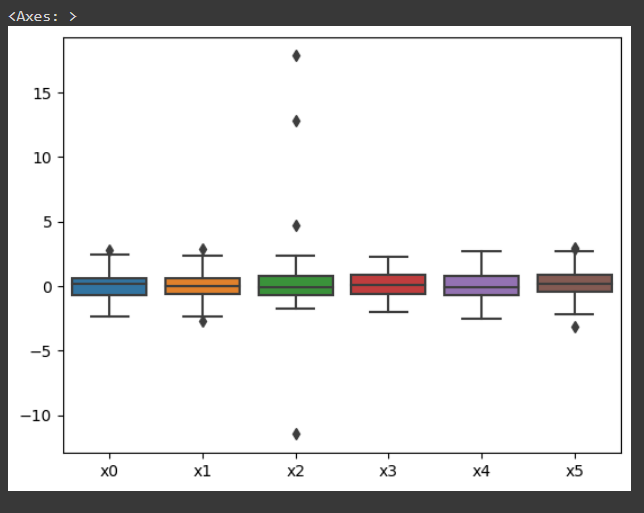


df = df.drop('x6', axis=1) # Слишком много пустых значений  
df = df.dropna() # обработка пропущенных значений  
  
df.info()  


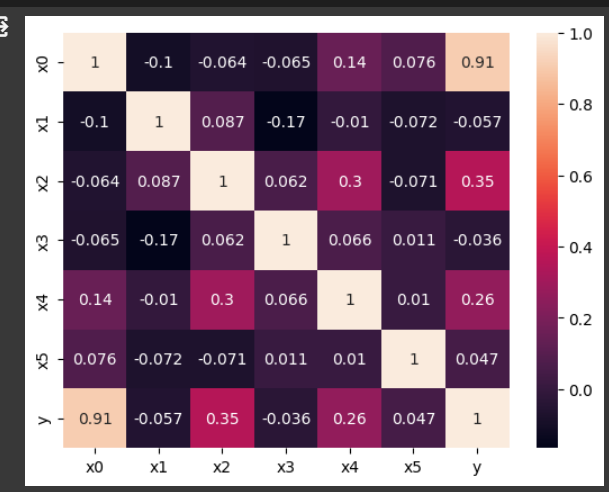
"""### 2. Проведите статистический анализ, оцените корреляции между признаками, оцените возможные выбросы в данных, постройте поясняющие графики. Удалите из рассмотрения признаки, которые не влияют на целевой.  
  
"""  
  
df.describe() # Есть какие-то выбросы в x2



sns.boxplot(data=df.drop('y', axis=1)) # Ящик для оценки выбросов



# матрица корреляции  
df = df.loc[df['x2'] < 4]  
df = df.loc[df['x2'] > -5]  
  
  
corr\_matrix = df.corr()  
sns.heatmap(corr\_matrix, annot=True)  
plt.show()  
  
# оценка корреляции



# Удаление неважных признаков  
df = df.drop('x1', axis=1)  
  
"""### 3. Разбиение выборки на тренировочные и тестовые данные (60% на 40%)."""  
  
X = df.drop('y', axis=1)  
y = df['y']  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=42)  
  
"""### 4. Для предсказания целевого признака постройте следующие модели, используя библиотеку sklearn:  
  
\* a. Линейная регрессия LinearRegression()  
\* b. Случайный лес RandomForestRegressor()  
\* c. Метод ближайших соседей KNeighborsRegressor()  
\* d. Метод опорных векторов SVR()  
  
#### a. Линейная регрессия LinearRegression()  
"""  
  
lr = LinearRegression()  
lr.fit(X\_train, y\_train)

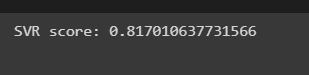
print("score LinearRegression()", lr.score(X\_test, y\_test))

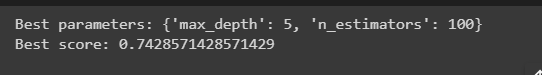
  
  
"""#### b. Случайный лес RandomForestRegressor()"""  
  
rf = RandomForestRegressor(random\_state=42)  
rf.fit(X\_train, y\_train)

print("Random Forest score:", rf.score(X\_test, y\_test))

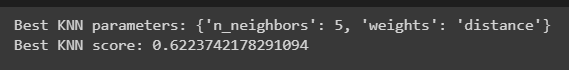
  
  
"""#### c. Метод ближайших соседей KNeighborsRegressor()"""  
  
knn = KNeighborsRegressor(n\_neighbors=5)  
knn.fit(X\_train, y\_train)  
  
print("KNN score:", knn.score(X\_test, y\_test))

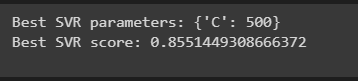
  
  
"""#### d. Метод опорных векторов SVR()"""  
  
svr = SVR(kernel='linear')  
svr.fit(X\_train, y\_train)  
print("SVR score:", svr.score(X\_test, y\_test))

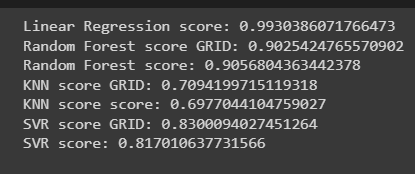
  
  
"""### 5. Для каждой модели попытайтесь добиться наилучшего возможного результата предсказания.  
  
#### Случайный лес  
"""  
  
param\_grid = {  
 'n\_estimators': [100, 200, 300],  
 'max\_depth': [5, 10, 15]  
}  
  
grid\_search\_rf = GridSearchCV(rf, param\_grid=param\_grid, cv=5)  
grid\_search\_rf.fit(X\_train, y\_train)  
  
print("Best parameters:", grid\_search\_rf.best\_params\_)  
print("Best score:", grid\_search\_rf.best\_score\_)



"""#### Метод ближайших соседей"""  
  
param\_grid = {  
 'n\_neighbors': [1, 2, 3, 5, 7, 9, 11],  
 'weights': ['distance']  
}  
  
# knn = KNeighborsRegressor()  
grid\_search\_knn = GridSearchCV(knn, param\_grid=param\_grid, cv=5)  
grid\_search\_knn.fit(X\_train, y\_train)  
  
print("Best KNN parameters:", grid\_search\_knn.best\_params\_)  
print("Best KNN score:", grid\_search\_knn.best\_score\_)

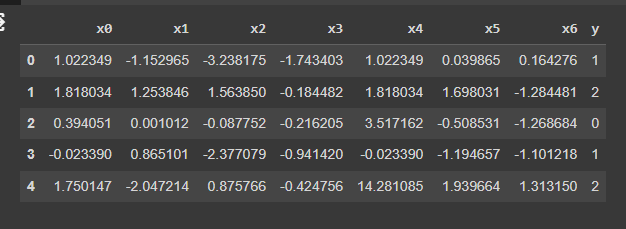
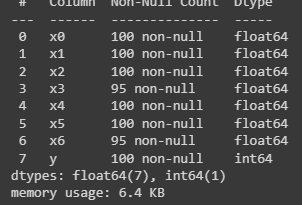
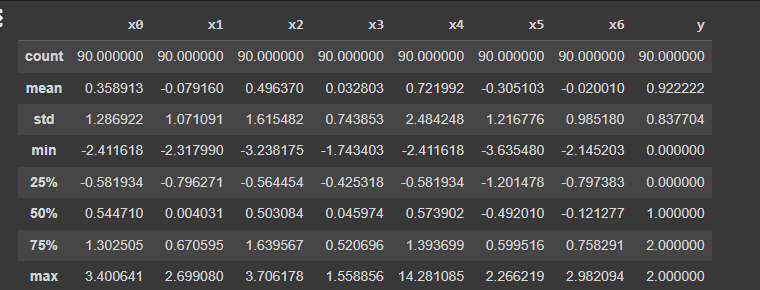
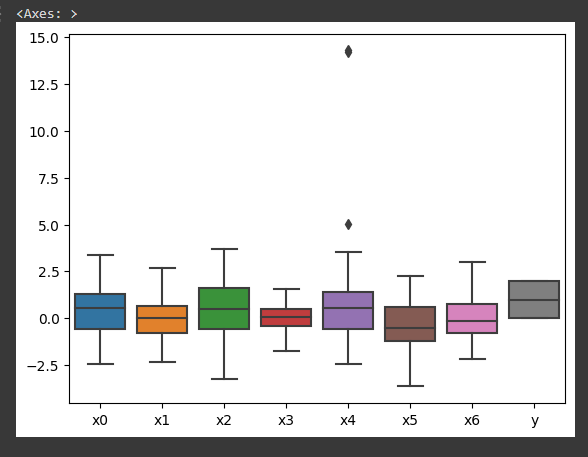
  
"""#### Метод SVR()"""  
  
param\_grid = {  
 'C': [0.01, 0.1, 1, 5, 10, 15, 50, 100, 150, 300, 400, 450, 500]  
}  
  
# svr = SVR()  
grid\_search\_svr = GridSearchCV(svr, param\_grid=param\_grid, cv=5)  
grid\_search\_svr.fit(X\_train, y\_train)  
print("Best SVR parameters:", grid\_search\_svr.best\_params\_)  
print("Best SVR score:", grid\_search\_svr.best\_score\_)

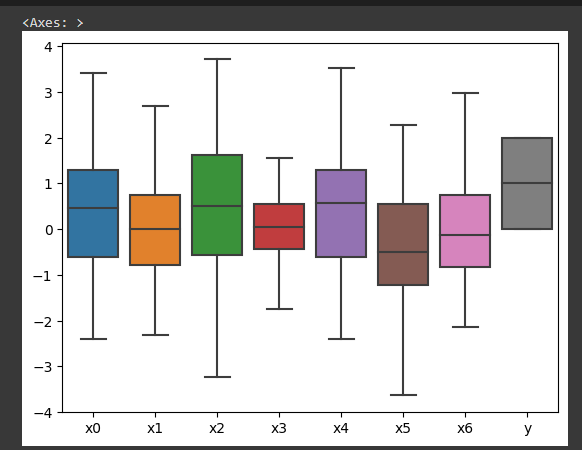
  
  
# Точность В методе SVR стала выше  
  
"""### 6. Оцените качество работы моделей. Выведите для каждой модели ее score(). Напишите какая модель дала наилучший результат. Выведите ее значение feature\_importances\_"""  
  
# Линейная регрессия  
print("Linear Regression score:", lr.score(X\_test, y\_test))  
  
# Случайный лес  
print("Random Forest score GRID:", grid\_search\_rf.score(X\_test, y\_test))  
print("Random Forest score:", rf.score(X\_test, y\_test))  
  
# Метод ближайших соседей  
print("KNN score GRID:", grid\_search\_knn.score(X\_test, y\_test))  
print("KNN score score:", knn.score(X\_test, y\_test))  
  
# Метод опорных векторов  
print("SVR score GRID:", grid\_search\_svr.score(X\_test, y\_test))  
print("SVR score:", svr.score(X\_test, y\_test))

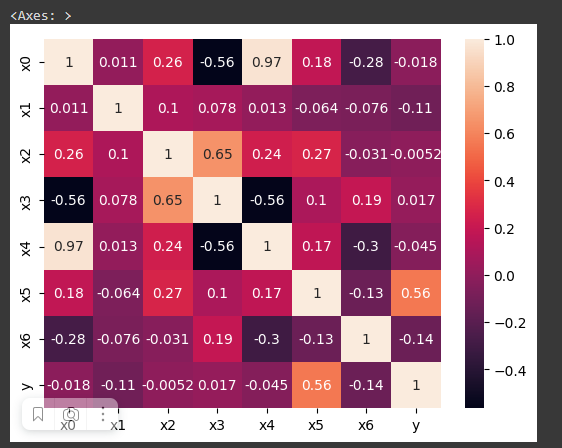
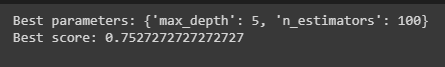


# Метод опорных векторов справился лучше всех  
  
svr = SVR(C=5, kernel='linear').fit(X\_train, y\_train)  
svr.score(X\_test, y\_test)

­  
  
print("SVR feature importances:", svr.coef\_)  
  
#Больше всего на целевой признак влияет x0 так как его вес самый большой  
  
"""## №2 Задача классификации (20 баллов)  
  
Возьмите набор данных согласно своему варианту  
(https://cloud.mail.ru/public/MWZh/DQkbAHKmi). Датасет представляет собой набор  
признаков, в последнем столбце указан класс объекта.  
"""  
  
df = pd.read\_csv("dataset\_var3\_class.csv", sep=';', names=["x0","x1","x2","x3","x4","x5","x6","y"])  
df.head()

  
"""### 1. Проведите предварительный анализ и подготовку данных, обработайте пропущенные значения."""  
  
df.info()  
  
# Удаление 0 значений  
df = df.dropna()  
  
df.info()  
  
"""### 2. Проведите статистический анализ, оцените корреляции между признаками, оцените возможные выбросы в данных, постройте поясняющие графики. Удалите из рассмотрения признаки, которые не влияют на целевой."""  
  
# Статистический анализ данных  
df.describe()  
  
# возможно x4 делает выбросы  
  
sns.boxplot(data = df)  
# х4 делает выбросы  
  
df = df.loc[df['x4'] < 5]  
sns.boxplot(data = df)

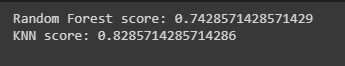
  
sns.heatmap(df.corr(), annot = True)

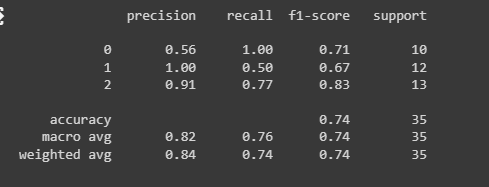
  
  
df = df.drop('x6', axis=1)  
  
"""### 3. Разбейте выборку на тренировочные и тестовые данные (60% на 40%)."""  
  
X = df.drop('y', axis=1)  
y = df['y']  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=42)  
  
"""### 4. Для предсказания класса постройте две модели, используя библиотеку sklearn:  
  
\* a. Случайный лес RandomForestClassifier()  
  
\* b. Метод ближайших соседей KNeighborsClassifier()  
  
#### a. Случайный лес RandomForestClassifier()  
"""  
  
rf = RandomForestClassifier(random\_state=42)  
rf.fit(X\_train, y\_train)  
  
rf.score(X\_test, y\_test)  
  
"""#### b. Метод ближайших соседей KNeighborsClassifier()"""  
  
knn = KNeighborsClassifier(n\_neighbors=5)  
knn.fit(X\_train, y\_train)  
  
knn.score(X\_test, y\_test)  
  
"""### 5. Для каждой модели попытайтесь добиться наилучшего возможного результата предсказания.  
  
#### a. Случайный лес RandomForestClassifier()  
"""  
  
param\_grid = {  
 'n\_estimators': [100, 200, 300],  
 'max\_depth': [5, 10, 15]  
}  
  
rf = RandomForestClassifier(random\_state=42)  
grid\_search\_rf = GridSearchCV(rf, param\_grid=param\_grid, cv=5)  
grid\_search\_rf.fit(X\_train, y\_train)  
  
print("Best parameters:", grid\_search\_rf.best\_params\_)  
print("Best score:", grid\_search\_rf.best\_score\_)  


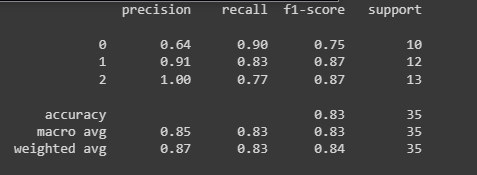
rf.fit(X\_train, y\_train)  
rf.score(X\_test, y\_test)  


# Точность увеличилась при подборе гиперпараметров в отличии от точности самой модели  
  
"""#### b. Метод ближайших соседей KNeighborsClassifier()"""  
  
param\_grid = {  
 'n\_neighbors': [1, 2, 3, 5, 7, 9, 11],  
 'weights': ['distance']  
}  
  
knn = KNeighborsClassifier()  
grid\_search\_knn = GridSearchCV(knn, param\_grid=param\_grid, cv=5)  
grid\_search\_knn.fit(X\_train, y\_train)  
  
print("Best KNN parameters:", grid\_search\_knn.best\_params\_)  
print("Best KNN score:", grid\_search\_knn.best\_score\_)  


knn.fit(X\_train, y\_train)  
  
knn.score(X\_test, y\_test)

  
# Здесь точночть выше у самой модели  
  
"""### 6. Оцените качество работы моделей. Выведите метрики модели. Напишите какая модель дала наилучший результат."""  
  
# Случайный лес  
print("Random Forest score:", grid\_search\_rf.score(X\_test, y\_test))  
  
# Метод ближайших соседей  
print("KNN score:", grid\_search\_knn.score(X\_test, y\_test))  


# у метода ближайших соседей точность выше  
  
# Случайный лес  
  
y\_pred = grid\_search\_rf.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))  
  
# Метрики хуже  
  
# Ближайшие соседи  
  
y\_pred = grid\_search\_knn.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))

  
# Все метрики близки к 1, что говорит о высоком качетсве модели