ГУАП

КАФЕДРА № 41

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

ассистент		В.В. Боженко
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

Отчет о лабораторной работе №4

Применение методов классификации

По курсу: Введение в анализ данных

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4917		Е.А. Ясиновский
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

Цель работы: изучить алгоритмы и методы классификации на практике.

Вариант:

.csv

Содержит информацию о сердечных болезнях.

1. Возраст
2. Пол
3. Общий билирубин
4. Прямой билирубин
5. Щелочная фосфатаза
6. АЛТ
7. АСТ
8. Уровень белков
9. Альбумин
10. Отношение альбумина к глобулину
11. Диагноз: Выходной класс [1: Болен: 0: Здоров]

Выполнение работы:

Была выполнена загрузка и очистка данных от явных и неявных дубликатов, были устранены пропуски в данных (Рисунок 1)

```
import pandas as pd
import numpy as np
import seaborn as sns
df = pd.read_csv("liver.csv")
df.duplicated().where(lambda x: x == True).dropna() ##  Явных дубликатов не найдено
df = df.dropna().reset_index()

# for col in df.columns:
#     print(df[col].unique())

df["Gender"] = df["Gender"].replace("Mal","Male")
df["Gender"] = df["Gender"].replace(["Male","Female"],[0,1])
df["Dataset123"] = df["Dataset123"].replace(["yes","no"],["1","2"])
df["Dataset123"] = df["Dataset123"].replace(["1","2"],["1","0"])
df["Aspartate_Aminotransferase"] = df["Aspartate_Aminotransferase"].replace("3a4","34")
df = df.rename(columns={"Dataset123":"IsSick"})
df = df.astype({
    'Gender':'int64',
    "IsSick":"int64",
    "Alkaline_Phosphotase":"int64",
    "Aspartate_Aminotransferase":"int64"})
```

✓ 0.6s

Рисунок 1 – Загрузка и очистка данных

Далее я проверил отношения объектов выходных классов, чтобы узнать подходит ли наша выборка под условия баланса классов (Рисунок 2)

```
isSick = list(df["IsSick"])
Sick = len(list(filter(lambda x: x == 1,isSick)))
ratio = Sick/len(isSick)
ratio
```

✓ 0.3s

```
0.7155172413793104
```

Рисунок 2 — Проверка условия баланса классов

Больных людей в нашей выборке 71%, а больных соответственно - 29%. Это отношение не соответствует условию баланса классов, значит ориентироваться на метрику Accuracy при выборе наиболее лучшей модели не стоит.

После проверки условия на баланс я подготовил данные для линейных моделей, а именно убрал показатели, корреляция которых слишком высока, и заменил их одним показателем, который показывает среднее значение среди двух зависимых показателей. Для этого я сначала нашел такие показатели (рисунок 3) и затем заменил их новым столбцом в датафрейме (рисунок 4)

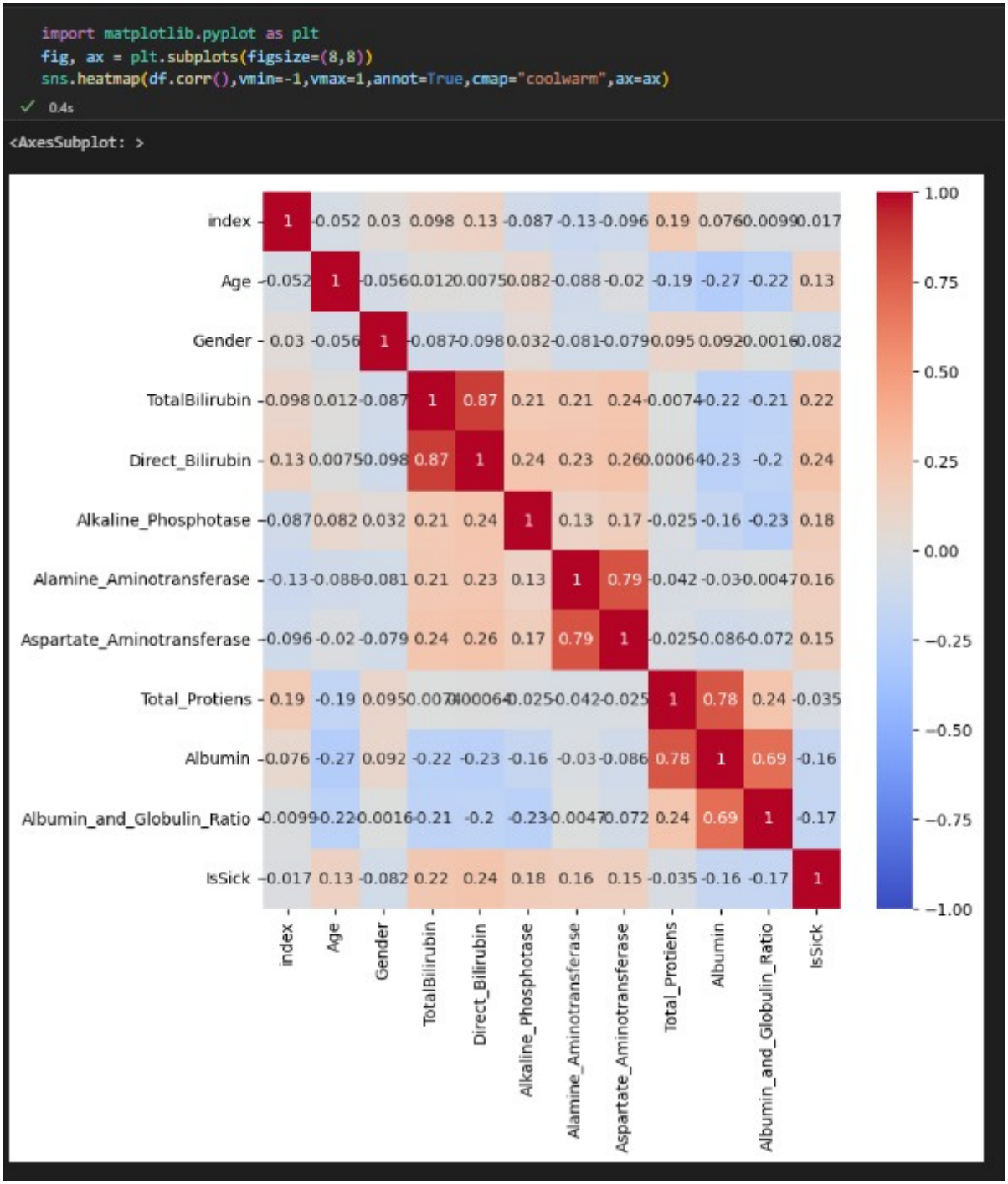

Рисунок 3 — Тепловая карта корреллляции

```
pairs = [["Alamine_Aminotransferase","Aspartate_Aminotransferase"],["Albumin","Total_Protiens"],["Direct_Bilirubin","TotalBilirubin"]]
AvgNames = ["ALT_AST","ALBUMINPROTEIN","BILIRUBIN"]
for i in range(len(pairs)):
    avg = [float(df[pairs[i][0]][j]+df[pairs[i][1]][j])/2 for j in range(len(df[pairs[i][0]]))]
    df[AvgNames[i]] = avg

for i in pairs:
    for j in i:
        df = df.drop([j],axis=1)
```

Рисунок 4 — Устранение зависимых показателей

После всех подготовительных и проверочных этапов я разделил наш датасет на 2 выборки — тренировочную и проверочную(рисунок 5). Затем создал три разные модели (K-соседей, Случайный лес, Логистическая регрессия), обучил их на тренировочных данных а затем сделал предсказания на основе проверочных данных и рассчитал их метрики(Рисунок 6).

```
from sklearn.model_selection import train_test_split
target = df["IsSick"]
data = df.drop(["IsSick"],axis=1)
dataTrain, dataTest, targetTrain,targetTest = train_test_split(data,target,test_size=0.5,random_state=0)
```
0.4s

Рисунок 5 — Разделение выборки

```
scaler = StandardScaler()
scaler.fit(data)
dataTrain = scaler.transform(dataTrain)
dataTest  = scaler.transform(dataTest)
#Обучение моделей
models={
"KNeighbors":KNeighborsClassifier(n_neighbors=5),
"RandomForest":RandomForestClassifier(n_estimators=100),
"LogisticRegression": LogisticRegression(max_iter=5000)}
predictions = {}
metrics = {}
for key in models:
    models[key].fit(dataTrain, targetTrain)
    predictions[key] = models[key].predict(dataTest)
    metrics[key] = {
        "ACC":accuracy_score(targetTest,predictions[key]),
        "PREC":precision_score(targetTest,predictions[key]),
        "RECL":recall_score(targetTest,predictions[key]),
        "BLAC":balanced_accuracy_score(targetTest,predictions[key]),
        "F1":f1_score(targetTest,predictions[key])
    }
```
0.1s

Рисунок 6 — Обучение моделей и расчет метрик

Сами метрики продемонстрированы на рисунке 7, помимо основных метрик я также сделал свою метрику, которая отображает среднее значение среди всех метрик, чтобы модели можно было оценить и по этому показателю.

```
for key in models:
    print(key,end=":\n")
    for metric in metrics[key]:
        print("\t",metric,":",str(metrics[key][metric]))
    print("\tAVG:", sum(metrics[key].values())/5)
```

✓ 0.3s

```
KNeighbors:
        ACC : 0.6293103448275862
        PREC : 0.6853932584269663
        RECL : 0.8026315789473685
        BLAC : 0.5513157894736842
        F1 : 0.7393939393939394
       AVG: 0.6816089822139089
RandomForest:
        ACC : 0.6724137931034483
        PREC : 0.6862745098039216
        RECL : 0.9210526315789473
        BLAC : 0.5605263157894737
        F1 : 0.7865168539325842
       AVG: 0.725356820841675
LogisticRegression:
        ACC : 0.6982758620689655
        PREC : 0.6880733944954128
        RECL : 0.9868421052631579
        BLAC : 0.5684210526315789
        F1 : 0.8108108108108107
       AVG: 0.7504846450539852
```

Рисунок 7 — Метрики моделей

Основываясь на предоставленных данных, мы можем сказать что все наши модели хорошо показывают себя на метрике Recall (>0.8), а в самой эффективной модели Logistic regression этот показатель даже выше 98%.

После оценки метрик был произведен расчет матрицы неточностей для каждой модели(рисунок 8).

```
from sklearn.metrics import confusion_matrix
cms = {}
for model in models:
    cm = confusion_matrix(targetTest,predictions[model])
    tn,fp,fn,tp = cm.ravel()
    cms[model] ={
        "TP":tp,
        "FP":fp,
        "TN":tn,
        "fn":fn
    }
cms
```

✓ 0.4s

```
{'KNeighbors': {'TP': 61, 'FP': 28, 'TN': 12, 'fn': 15},
 'RandomForest': {'TP': 70, 'FP': 32, 'TN': 8, 'fn': 6},
 'LogisticRegression': {'TP': 75, 'FP': 34, 'TN': 6, 'fn': 1}}
```

Рисунок 8 — Матрица неточносней

Из данной матрицы видно, что наши модели оказались "оптимистичными" и в них достаточно много ложно-положительных результатов, однако можно заметить, что у метода к-сосеседей также и достаточно большое количество ложно-отрицательных результатов.

После расчета матрицы неточностей я произвел расчет ROC кривых для наших моделей (рисунок 9) и отобразил график, который отображает эти кривые (рисунок 10).

```
from matplotlib import pyplot as plt
rocs = {}
for model in models:
    prob = models[model].predict_proba(dataTest)
    prob = prob[:,1]
    model_auc = roc_auc_score(targetTest,prob)
    fpr, tpr, threshold = roc_curve(targetTest,prob)
    roc_auc = auc(fpr,tpr)
    rocs[model]={
        "TPR":tpr,
        "FPR":fpr,
        "AUC":roc_auc
    }
for roc in rocs:
    print(roc," ROC AUC:", str(rocs[roc]["AUC"]))
```

✓ 0.4s

```
KNeighbors  ROC AUC: 0.6600328947368421
RandomForest  ROC AUC: 0.7516447368421052
LogisticRegression  ROC AUC: 0.7628289473684211
```

Рисунок 9 - Расчет ROC кривых

```
plt.figure(figsize=(7,7))
for model in models:
    plt.plot(rocs[model]["FPR"],rocs[model]["TPR"],label=model % rocs[model]["AUC"])
plt.plot([0,1],[0,1],color="black",linestyle="--")
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.01])
plt.xlabel("FP Rate")
plt.ylabel("TP Rate")
plt.title("ROC кривые")
plt.legend(loc="lower right")
plt.show()
```

✓ 0.3s

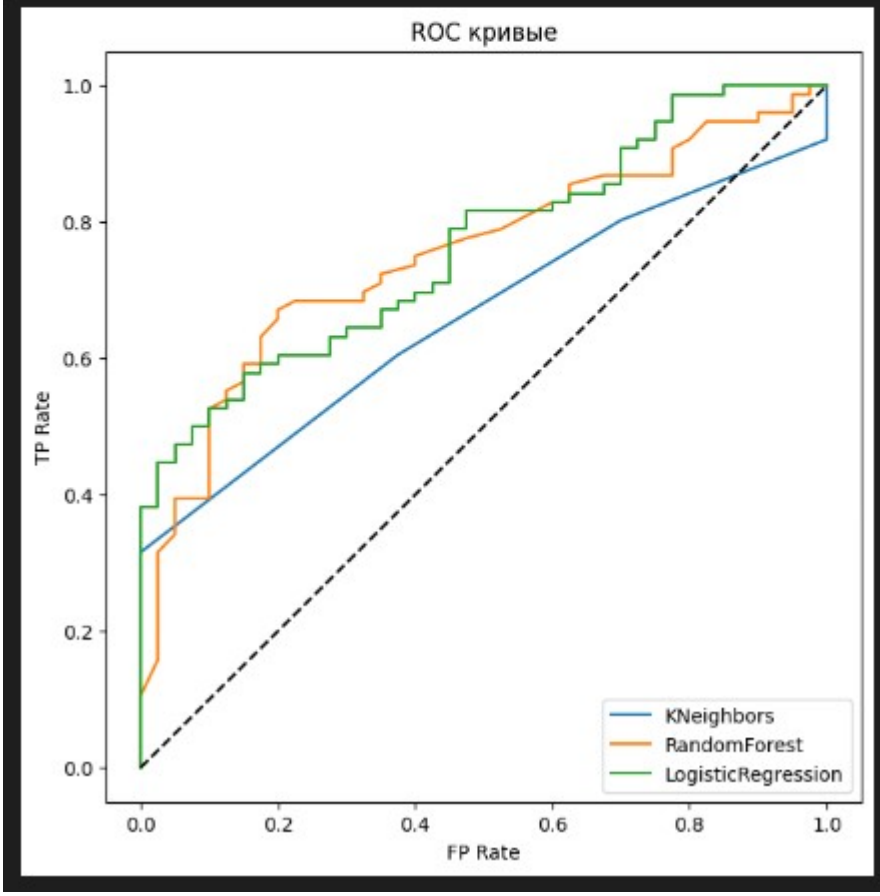

Рисунок 10 — График ROC кривых

Ссылка на Jupyter notebook: https://github.com/EgorYasinovskiy/Data-Analys.git

Вывод:

Во время выполнения данной лабораторной работы были получены практические навыки использования алгоритмов и методов классификации, таких как метод ближайших соседей, логическая регрессия и случайное дерево. После оценки классификации по различным метрикам, построения матрицы неточности и визуализации графика ROC кривых, можно сделать вывод, что несмотря на некоторые недостатки, наилучший результат был получен методом логистической регрессии, так как в среднем результаты этого метода лучше, чем у остальных моделей.