Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Автоматизированные системы обработки информации и управления»

Отчет

Лабораторная работа № 4

По курсу «Технологии машинного обучения»

ИСПОЛНИТЕЛЬ:
Группа ИУ5-65Б
Голубев С.Н.

"24" мая 2021 г.

ПРЕПОДАВАТЕЛЬ:
Гапанюк Ю.Е.

_____

"__"_____2021 г.

Москва 2021

1. Задание

Выберите набор данных (датасет) для решения задачи классификации или регрессии. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков. С использованием метода train_test_split разделите выборку на обучающую и тестовую. Обучите следующие модели:

- одну из линейных моделей;
- SVM;
- дерево решений.
- Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.

2. Скрины jupyter notebook

ЛР4 Голубев Сергей ИУ5-65Б

В [1]:
```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.impute import SimpleImputer
import warnings
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score, classification_report
from sklearn.linear_model import LinearRegression
warnings.simplefilter("ignore")
```

В [2]:
```
# чтение обучающей выборки
data = pd.read_csv('diabetes.csv')
data.head()
```

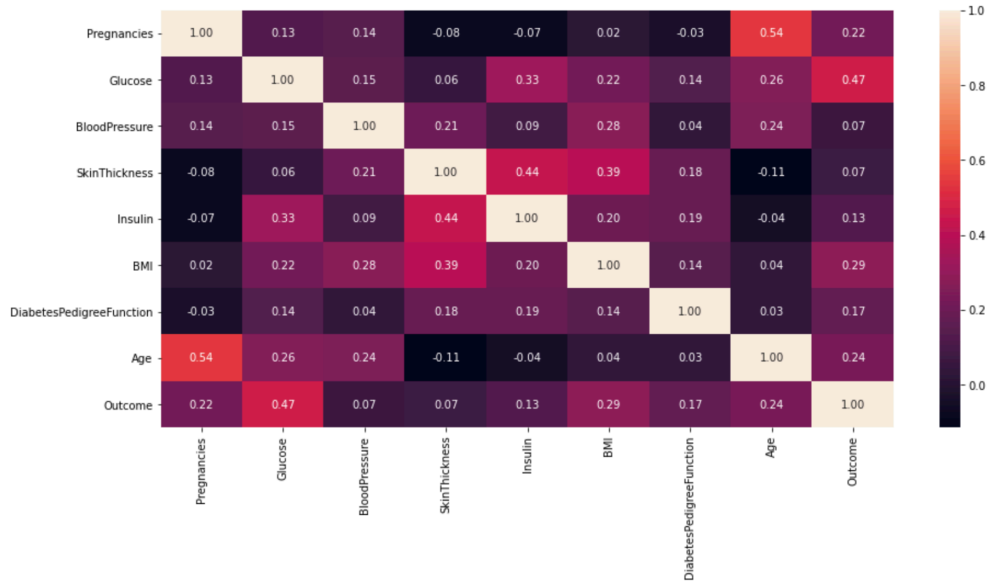
Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

В [3]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

В [4]:
```
#Построим корреляционную матрицу
fig, ax = plt.subplots(figsize=(15,7))
sns.heatmap(data.corr(method='pearson'), ax=ax, annot=True, fmt='.2f')
```

Out[4]: <AxesSubplot:>



В [5]:
```
X = data[["Age","Outcome"]]
Y = data.Pregnancies
print('Входные данные:\n\n', X.head(), '\n\nВыходные данные:\n\n', Y.head())
```

Входные данные:

```
    Age  Outcome
0   50         1
1   31         0
2   32         1
3   21         0
4   33         1
```

Выходные данные:

```
 0    6
1    1
2    8
3    1
4    0
Name: Pregnancies, dtype: int64
```

В [6]:
```
X_train,  X_test,  Y_train,  Y_test = train_test_split(X,  Y, random_state = 0, test_size = 0.1)
print('Входные параметры обучающей выборки:\n\n',X_train.head(), \
      '\n\nВходные параметры тестовой выборки:\n\n', X_test.head(), \
      '\n\nВыходные параметры обучающей выборки:\n\n', Y_train.head(), \
      '\n\nВыходные параметры тестовой выборки:\n\n', Y_test.head())
```

Входные параметры обучающей выборки:

```
      Age  Outcome
499   39         0
720   34         0
556   30         0
583   42         0
150   24         0
```

Входные параметры тестовой выборки:

```
      Age  Outcome
661   22         1
122   23         0
113   25         0
14    51         1
529   31         0
```

Выходные параметры обучающей выборки:

```
 499    6
720    4
556    1
583    8
150    1
Name: Pregnancies, dtype: int64
```

Выходные параметры тестовой выборки:

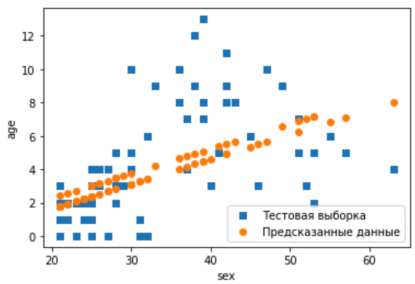
```
 661    1
122    2
113    4
14     5
529    0
Name: Pregnancies, dtype: int64
```

В [7]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error,  median_absolute_error, r2_score
```

В [8]:
```
Lin_Reg = LinearRegression().fit(X_train, Y_train)

lr_y_pred = Lin_Reg.predict(X_test)
```

В [9]:
```
plt.scatter(X_test.Age, Y_test,    marker = 's', label = 'Тестовая выборка')
plt.scatter(X_test.Age, lr_y_pred, marker = 'o', label = 'Предсказанные данные')
plt.legend (loc = 'lower right')
plt.xlabel ('sex')
plt.ylabel ('age')
plt.show()
```



SVM

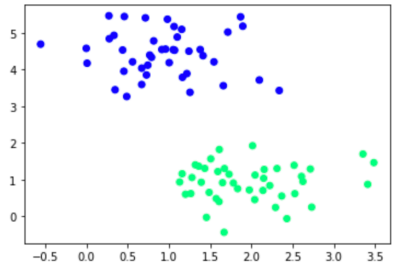
В [10]:
```
from sklearn.svm import SVC , LinearSVC
from sklearn.datasets.samples_generator import make_blobs
from matplotlib import pyplot as plt
```

B [11]:
```
X , y = make_blobs(n_samples=125, centers=2, cluster_std=0.6, random_state=0)
# колич, кол кластеров, станд откл,

train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=40, random_state=0)

plt.scatter(train_X[:, 0], train_X[:, 1], c=train_y, cmap='winter')
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0x7fafd18b1910>
```



B [12]:
```
svc = SVC(kernel='linear')
svc.fit(train_X,train_y)
```

Out[12]:
```
SVC(kernel='linear')
```

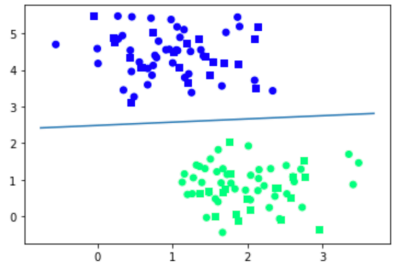
B [13]:
```
plt.scatter(train_X[:, 0], train_X[:, 1], c=train_y, cmap='winter')

ax=plt.gca()
xlim=ax.get_xlim()

ax.scatter(test_X[:, 0], test_X[:, 1], c=test_y, cmap='winter', marker='s')

w= svc.coef_[0]
a= -w[0]/w[1]
xx=np.linspace(xlim[0], xlim[1])
yy= a * xx - (svc.intercept_[0]/ w[1])
plt.plot(xx, yy)
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x7fafd1a79bb0>]
```



B [14]:
```
pred_y = svc.predict(test_X)
```

B [15]:
```
confusion_matrix(test_y, pred_y)
```

Out[15]:
```
array([[21,  0],
       [ 0, 19]])
```

Tree

B [16]:
```
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz
from sklearn.tree import export_graphviz
from sklearn import tree
import re
```

In [17]:
```
X = data[["Age","Outcome"]]
Y = data.Pregnancies
print('Входные данные:\n\n', X.head(), '\n\nВыходные данные:\n\n', Y.head())
```

Входные данные:

```
    Age  Outcome
0   50         1
1   31         0
2   32         1
3   21         0
4   33         1
```

Выходные данные:

```
 0    6
1    1
2    8
3    1
4    0
Name: Pregnancies, dtype: int64
```

In [18]:
```
# Обучим дерево на всех признаках iris
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, Y)
```

In [19]:
```
from IPython.core.display import HTML
from sklearn.tree.export import export_text
tree_rules = export_text(clf, feature_names=list(X.columns))
HTML('<pre>' + tree_rules + '</pre>')
```

Out[19]:
```
|--- Age <= 29.50
|   |--- Outcome <= 0.50
|   |   |--- Age <= 25.50
|   |   |   |--- Age <= 24.50
|   |   |   |   |--- Age <= 21.50
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- Age >  21.50
|   |   |   |   |   |--- Age <= 22.50
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- Age >  22.50
|   |   |   |   |   |   |--- Age <= 23.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- Age >  23.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |--- Age >  24.50
|   |   |   |   |--- class: 2
```

In [20]:
```
tree.plot_tree(clf)
```

Out[20]:
```
[Text(100.73934375, 210.645, 'X[0] <= 29.5\ngini = 0.894\nsamples = 768\nvalue = [111, 135, 103, 75, 68, 57, 50, 45,
38, 28, 24\n11, 9, 10, 2, 1, 1]'),
 Text(37.2, 197.055, 'X[1] <= 0.5\ngini = 0.808\nsamples = 396\nvalue = [87, 104, 88, 51, 32, 15, 14, 2, 1, 1, 1, 0,
0\n0, 0, 0, 0]'),
 Text(16.368, 183.465, 'X[0] <= 25.5\ngini = 0.793\nsamples = 312\nvalue = [61, 94, 75, 34, 25, 9, 12, 1, 0, 0, 1, 0,
0\n0, 0, 0, 0]'),
 Text(8.928, 169.875, 'X[0] <= 24.5\ngini = 0.755\nsamples = 222\nvalue = [49, 75, 58, 23, 12, 2, 2, 1, 0, 0, 0, 0, 0
\n0, 0, 0, 0]'),
 Text(5.952, 156.285, 'X[0] <= 21.5\ngini = 0.747\nsamples = 188\nvalue = [42, 68, 46, 18, 10, 1, 2, 1, 0, 0, 0, 0, 0
\n0, 0, 0, 0]'),
 Text(2.976, 142.695, 'gini = 0.706\nsamples = 58\nvalue = [18, 22, 13, 3, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0]'),
 Text(8.928, 142.695, 'X[0] <= 22.5\ngini = 0.759\nsamples = 130\nvalue = [24, 46, 33, 15, 8, 1, 2, 1, 0, 0, 0, 0, 0,
0\n0, 0, 0]'),
 Text(5.952, 129.10500000000002, 'gini = 0.742\nsamples = 61\nvalue = [13, 20, 18, 8, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0\n
0, 0, 0]'),
 Text(11.904, 129.10500000000002, 'X[0] <= 23.5\ngini = 0.766\nsamples = 69\nvalue = [11, 26, 15, 7, 6, 1, 2, 1, 0,
0, 0, 0, 0, 0\n0, 0, 0]'),
 Text(8.928, 115.515, 'gini = 0.726\nsamples = 31\nvalue = [6, 13, 7, 2, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0]'),
 Text(14.879999999999999, 115.515, 'gini = 0.791\nsamples = 38\nvalue = [5, 13, 8, 5, 4, 1, 1, 1, 0, 0, 0, 0, 0, 0\n
```