МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)

Кафедра прикладной математики

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 5

по дисциплине «Технологии и инструментарий анализа больших данных»

Выполнил студент группы ИКБО-20-21 Фомичев Р.А.

Проверил ассистент кафедры ПМ ИИТ Тетерин Н.Н.

Москва 2024

Задание № 1

Найти данные для классификации. Данные в группе повторяться не должны. Предобработать данные, если это необходимо.

Код программы представлен на рисунке 1.

```
df=pd.read_csv('sample_data/data.csv')
df.head()
```

Рисунок 1 – Код программы

Вывод программы представлен на рисунке 2.

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430

Рисунок 2 – Вывод программы

Задание № 2

Изобразить гистограмму, которая показывает баланс классов. Сделать выводы.

Код программы представлен на рисунке 3.

```
balance = df['diagnosis'].value_counts()

print(balance)

plt.bar(balance.index, balance.values)
plt.xlabel('Диагноз')
plt.ylabel('Количество записей')
plt.title('Баланс классов')
plt.xticks(balance.index, ['B', 'M'])
plt.show()
```

Рисунок 3 – Код программы

Вывод программы представлен на рисунке 4.

```
diagnosis
B    357
M    212
Name: count, dtype: int64
```

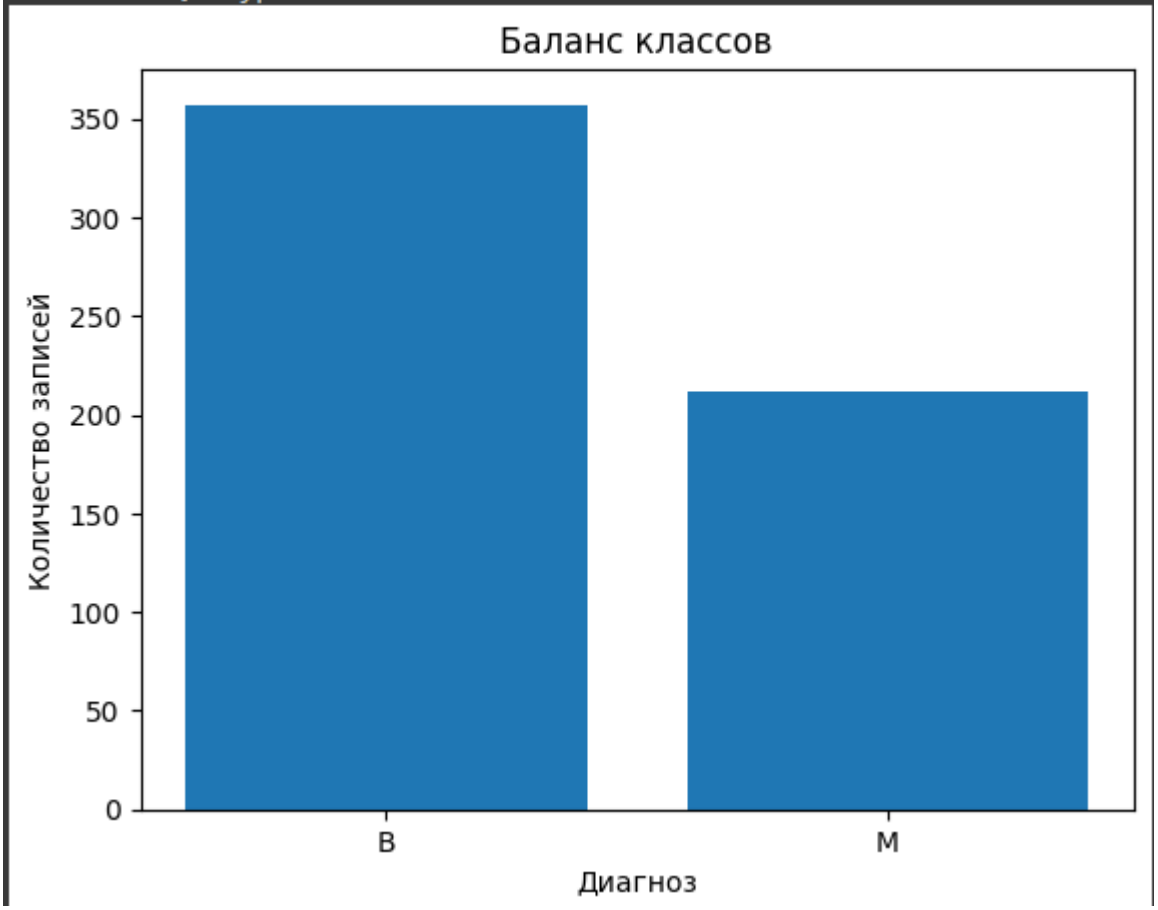

Рисунок 4 – Вывод программы

Задание № 3

Разбить выборку на тренировочную и тестовую. Тренировочная для обучения модели, тестовая для проверки ее качества.

Код программы представлен на рисунке 5.

```
target = df.iloc[:, 1]
predictors = df.iloc[:, 2:-1]

x_train, x_test, y_train, y_test = train_test_split(predictors, target, train_size=0.2, shuffle=True, random_state=42)

print('Размер для признаков обучающей выборки', x_train.shape)
print('Размер для признаков тестовой выборки', x_test.shape)
print('Размер для целевого показателя обучающей выборки', y_train.shape)
print('Размер для показателя тестовой выборки', y_test.shape)
```

Рисунок 5 – Код программы

Вывод программы представлен на рисунке 6.

```
Размер для признаков обучающей выборки (113, 30)
Размер для признаков тестовой выборки (456, 30)
Размер для целевого показателя обучающей выборки (113,)
Размер для показателя тестовой выборки (456,)
```

Рисунок 6 – Вывод программы

Задание № 4

Применить алгоритмы классификации: логистическая регрессия, SVM, KNN. Построить матрицу ошибок по результатам работы моделей (использовать confusion_matrix из sklearn.metrics)

Код программы представлен на рисунке 7.

```
print('Исходные значения: ', np.array(y_test))

#Logistic regression
print("Logistic Regression:")
model = LogisticRegression()
model.fit(x_train, y_train)
y_predict = model.predict(x_test)
print('Предсказанные значения: ', y_predict)

# Матрица ошибок
plt.rcParams['figure.figsize'] = (10, 10)
fig = px.imshow(confusion_matrix(y_test, y_predict), text_auto=True)
fig.update_layout(xaxis_title="Target", yaxis_title="Prediction")
fig.show()

#SVC
print("SVM:")
param_kernel = ('linear', 'rbf', 'poly', 'sigmoid') # для перебора ядер
parameters = {'kernel': param_kernel}
model = SVC()
grid_search_svm = GridSearchCV(estimator = model, param_grid = parameters, cv = 6) # сетка для перебора параметров
grid_search_svm.fit(x_train, y_train) # обучаем модели с разными параметрами
best_model = grid_search_svm.best_estimator_
print(best_model.kernel)
svm_preds = best_model.predict(x_test)
print('Предсказанные значения: ', svm_preds)

# Матрица ошибок SVM
plt.rcParams['figure.figsize'] = (10, 10)
fig = px.imshow(confusion_matrix(y_test, svm_preds), text_auto=True)
fig.update_layout(xaxis_title="Target", yaxis_title="SVM Prediction")
fig.show()

# KNN
print("KNN:")
number_of_neighbors = np.arange(3, 10) # количество соседей для перебора
model_KNN = KNeighborsClassifier() # инициализация модели
params = {"n_neighbors": number_of_neighbors}
grid_search = GridSearchCV(estimator=model_KNN, param_grid=params, cv=6)    # задание параметров для поиска по сетке
grid_search.fit(x_train, y_train) # обучение модели
print(grid_search.best_score_) # лучшее значение macro-average
print(grid_search.best_estimator_) # лучшая модель получается при k = 3
knn_preds = grid_search.predict(x_test) # результат работы модели для тестовых данных
print('Предсказанные значения: ',knn_preds)

# Матрица ошибок KNN
plt.rcParams['figure.figsize'] = (10, 10)
fig = px.imshow(confusion_matrix(y_test, knn_preds), text_auto=True)
fig.update_layout(xaxis_title="Target", yaxis_title="KNN Prediction")
fig.show()
```

Рисунок 7 – Код программы

Вывод программы представлен на рисунках 8 – 10.

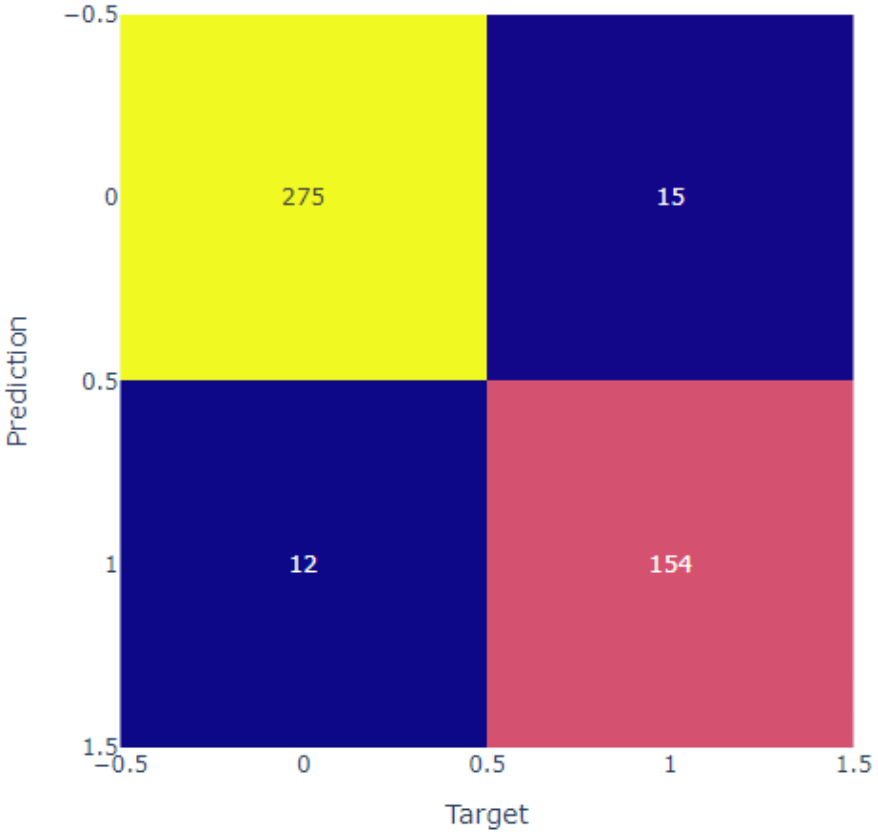

Рисунок 8 –Logistic Regression

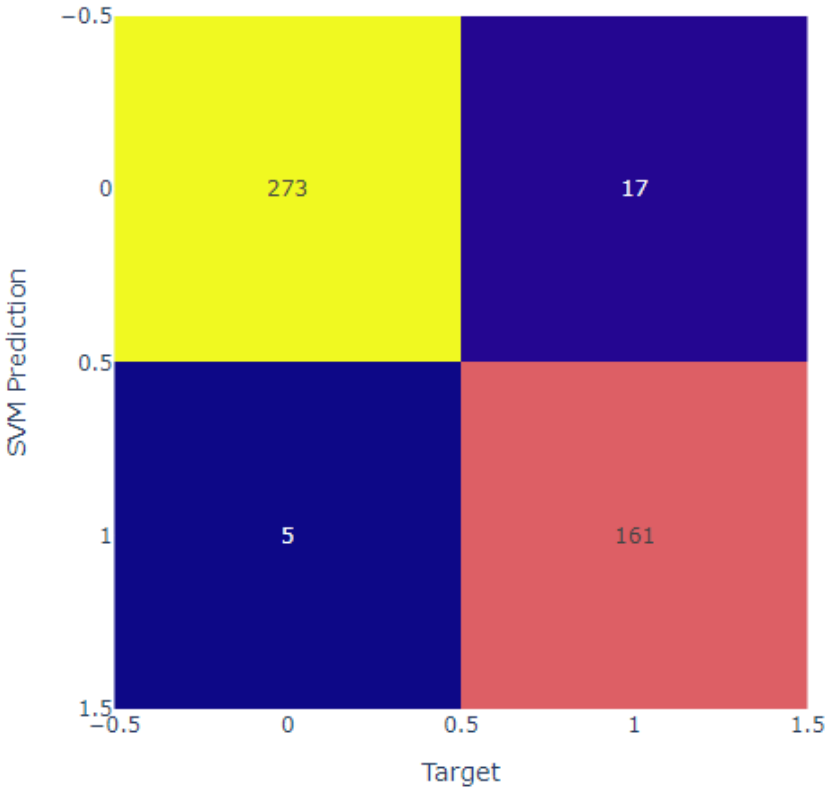

Рисунок 9 – SVM

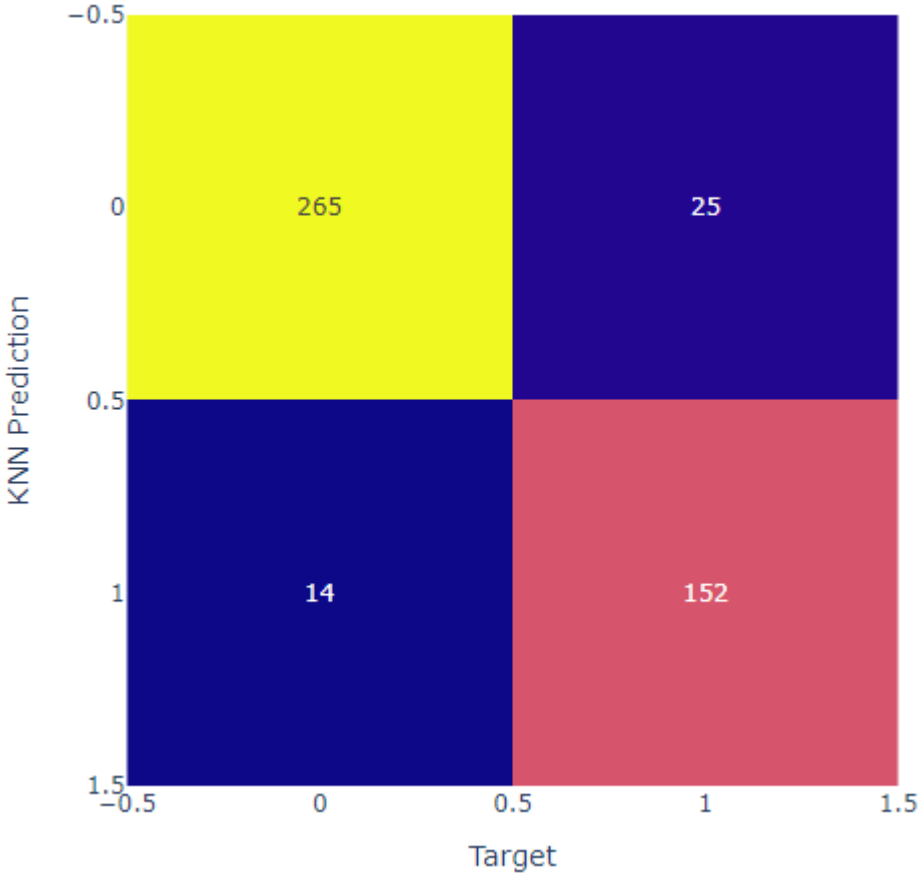

Рисунок 10 – KNN

Задание № 5

Сравнить результаты классификации, используя accuracy, precision, recall и f1-меру (можно использовать classification_report из sklearn.metrics). Сделать выводы.

Код представлен на рисунке 11.

```
print(classification_report(y_predict, y_test))
print(classification_report(svm_preds, y_test))
print(classification_report(knn_preds, y_test))
```

Вывод программы представлен на рисунке 12.

```
              precision    recall  f1-score   support

           B       0.95      0.96      0.95       287
           M       0.93      0.91      0.92       169

    accuracy                           0.94       456
   macro avg       0.94      0.93      0.94       456
weighted avg       0.94      0.94      0.94       456

              precision    recall  f1-score   support

           B       0.94      0.98      0.96       278
           M       0.97      0.90      0.94       178

    accuracy                           0.95       456
   macro avg       0.96      0.94      0.95       456
weighted avg       0.95      0.95      0.95       456

              precision    recall  f1-score   support

           B       0.91      0.95      0.93       279
           M       0.92      0.86      0.89       177

    accuracy                           0.91       456
   macro avg       0.91      0.90      0.91       456
weighted avg       0.91      0.91      0.91       456
```

Рисунок 12 – Вывод программы