Задание 1.
Исходные данные:
Измените функцию predict(w, X) так, чтобы можно было подать порог для классификации.
Решение:

```
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> import scipy
>>> import sklearn
>>> from sklearn.datasets import make_classification
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.metrics import accuracy_score, confusion_matrix, recall_score, roc_auc_score,
precision_score
>>> X, y = make_classification(n_classes=2, class_sep=1.5, weights=[0.9, 0.1], n_features=20,
n_samples=1000, random_state=10)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
>>> regressor = LogisticRegression(class_weight="balanced")
>>> regressor.fit(X_train, y_train)
LogisticRegression(C=1.0, class_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
>>> W = 0.25
>>> preds = np.where(regressor.predict_proba(X_test)[:,1] > W, 1, 0)
>>> pd.DataFrame(data=[accuracy_score(y_test, preds), recall_score(y_test, preds),
...                    precision_score(y_test,preds), roc_auc_score(y_test, preds)],
...             index=["accuracy", "recall", "precision", "roc_auc_score"])
                    0
accuracy       0.933333
recall         0.861111
precision      0.645833
roc_auc_score  0.901644
>>>
```

Задание 2.
Исходные данные:
Подберите аргументы функции optimize для логистической регрессии таким образом, чтобы log loss был минимальным
Решение:

```
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> import scipy
>>> import sklearn
>>> from sklearn.datasets import make_classification
>>> from sklearn.model_selection import train_test_split
```

```
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.metrics import accuracy_score, confusion_matrix, recall_score, roc_auc_score,
precision_score
>>> X, y = make_classification(n_classes=2, class_sep=1.5, weights=[0.9, 0.1], n_features=20,
n_samples=1000, random_state=10)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
>>> regressor = LogisticRegression(class_weight="balanced")
>>> regressor.fit(X_train, y_train)
LogisticRegression(C=1.0, class_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
>>> W = 0.1
>>> preds = np.where(regressor.predict_proba(X_test)[:,1] > W, 1, 0)
>>> pd.DataFrame(data=[accuracy_score(y_test, preds), recall_score(y_test, preds),
...                    precision_score(y_test,preds), roc_auc_score(y_test, preds)],
...              index=["accuracy", "recall", "precision", "roc_auc_score"])
                     0
accuracy      0.918182
recall        0.916667
precision     0.578947
roc_auc_score  0.917517
>>>
```

Задание 3.
Исходные данные:
Посчитайте Accuracy, матрицу ошибок, точность и полноту, а также F1 score.
Решение:

```
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> import scipy
>>> import sklearn
>>> from sklearn.datasets import make_classification
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.metrics import accuracy_score, confusion_matrix, recall_score, roc_auc_score,
precision_score, f1_score
>>> X, y = make_classification(n_classes=2, class_sep=1.5, weights=[0.9, 0.1], n_features=20,
n_samples=1000, random_state=10)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
>>> regressor = LogisticRegression(class_weight="balanced")
>>> regressor.fit(X_train, y_train)
LogisticRegression(C=1.0, class_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
>>> W = 0.25
```

```
>>> y_preds = np.where(regressor.predict_proba(X_test)[:,1] > W, 1, 0)
>>> y_true = preds = np.where(regressor.predict_proba(X_test)[:,1] > W, 1, 0)
>>> accuracy_score(y_true, y_preds)
1.0
>>> accuracy_score(y_true, y_preds, normalize=False)
330
>>> confusion_matrix(y_true, y_preds)
array([[282,   0],
       [  0,  48]])
>>> confusion_matrix(y_true, y_preds, normalize='all')
array([[0.85454545, 0.        ],
       [0.        , 0.14545455]])
>>> tn, fp, fn, tp = confusion_matrix(y_true, y_preds).ravel()
>>> tn, fp, fn, tp
(282, 0, 0, 48)
>>> precision_score(y_true, y_preds)
1.0
>>> recall_score(y_true, y_preds)
1.0
>>> f1_score(y_true, y_preds)
1.0
>>>
```