Вариант 1

```
def genData(size=500):
    data = np.random.rand(size, 2)*2 - 1
    label = np.zeros([size, 1])
    for i, p in enumerate(data):
        if (p[0] + .5 >= p[1]) and (p[0] - 0.5 <= p[1]):
            label[i] = 1.
        else:
            label[i] = 0.
    div = round(size*0.8)
    train_data = data[:div, :]
    test_data = data[div:, :]
    train_label = label[:div, :]
    test_label = label[div:, :]
    return (train_data, train_label), (test_data, test_label)
```

Функция генерирует координаты точек и разделяет их на две группы. Первая – точки, для которых выполняется $x - 0.5 \geq y \geq x + 0.5$, вторая – точки, для которых не выполняется. Значения координат находятся в диапазоне от -1 до 1.

Реализация модели

Выбрана последовательная модель.

```
model = models.Sequential()
```

Для модели выбрана следующая архитектура:

- 2 полносвязных скрытых слоя с 64 нейронами
- 1 входной слой с 1 нейроном, так как в результате необходим получить скаляр
- На полносвязных слоях используется функция активации relu
- На выходном слое будет использована сигмоидная функция. Значения можем интерпретировать как вероятность того, что точка принадлежит первому классу.

```
model.add(layers.Dense(64, activation='relu', input_shape=(2,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

В качество оптимизатора выбран RMSProp , функцией потерь бинарная кросс-энтропия (так как бинарная классификация), а в качестве метрики используется точность (процент верных предсказаний).

```
model.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['accuracy'])
```

Данные для обучения разделим так:
80% – данные, на которых обучается сеть;
20% – контрольный набор для проверки;

```
x_val = train_data[:len(train_data) * 2 // 10]
partial_x_train = train_data[len(train_data) * 2 // 10:]
y_val = train_label[:len(train_label) * 2 // 10]
partial_y_train = train_label[len(train_label) * 2 // 10:]
```

Обучение модели будет проходить на протяжении 100 эпох пакетами по 100 образцов.

```
H = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=100,
                    batch_size=len(partial_x_train) // 4,
                    validation_data=(x_val, y_val))
```

В ходе тестирования процент верных предсказаний на тестовых данных колебался от 97% до 100%, значение функции потерь изменялось от 0.04 до 0.09.

Графики работы модели

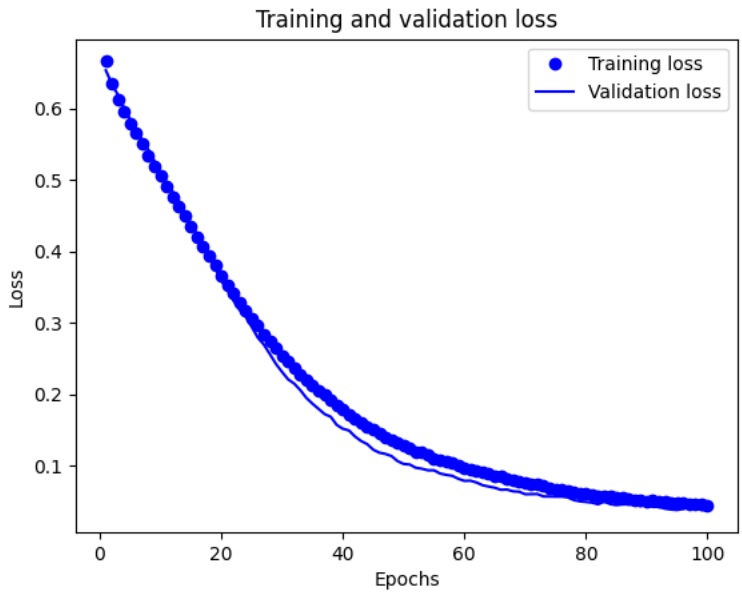

Рисунок 1 – График ошибки

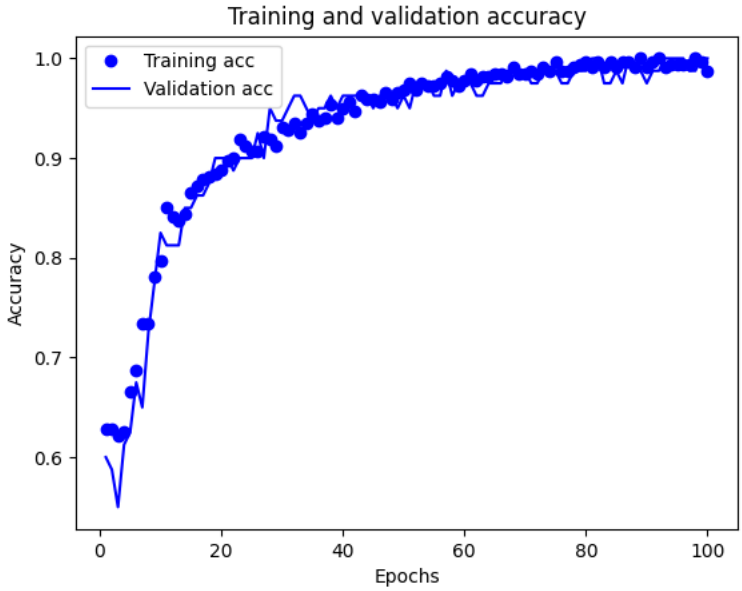

Рисунок 2 – График точности

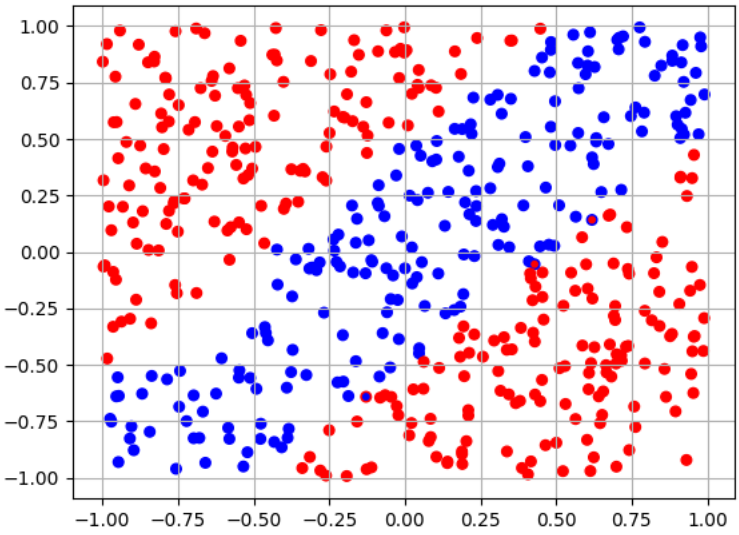

Рисунок 3 – Бинарная классификация

Из рис. 3 видно, что ошибочные предсказания близки к границам классов.