Был выполнен 4 вариант работы.

Функция для генерации данных поставляет два класса точек:

1) x = p * cos(2 * pi * p) + (r – 1)/(2*p)

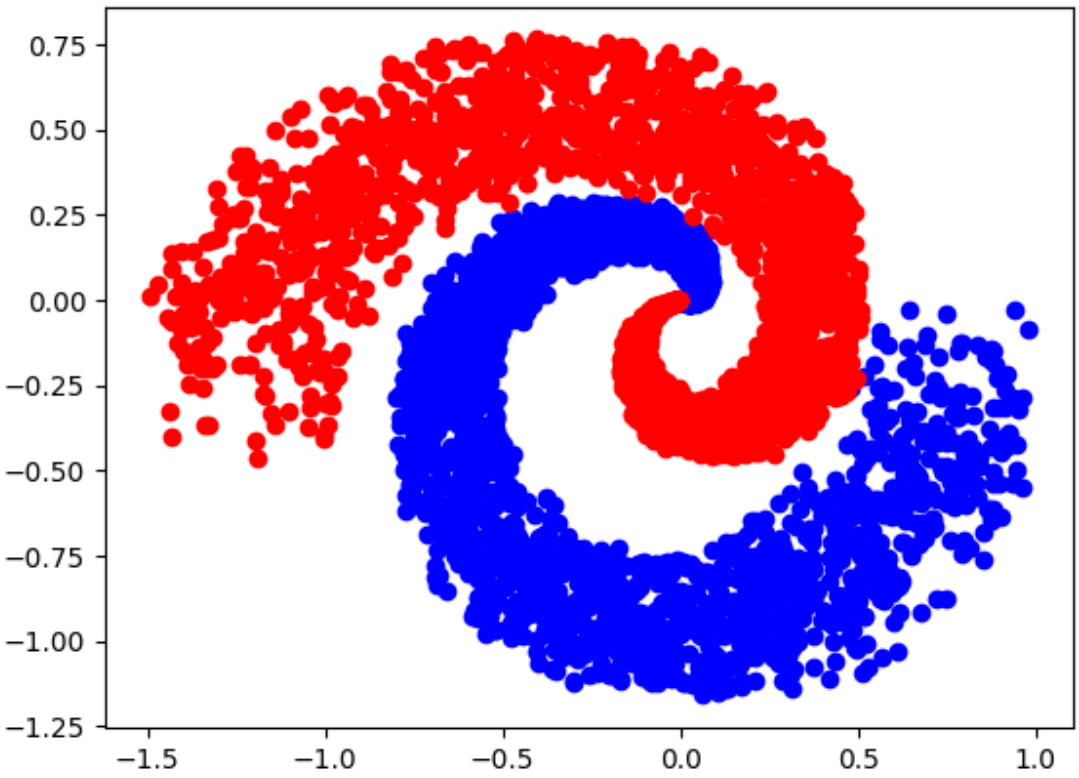
y = p * sin(2 * pi * p) + (r – 1)/(2*p)

Параметры r, p принадлежат [0, 1]

2) x = -p * cos(2 * pi * p) + (r – 1)/(2*p)

y = -p * sin(2 * pi * p) + (r – 1)/(2*p)

Визуализация двух классов:




```
model = models.Sequential()
model.add(layers.Dense(32, activation='relu', input_shape=(2,)))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

Была создана модель. Первый слой принимает 2 параметра (x, y). Первые 2 слоя состоят из 32 нейронов (опытным путем дало лучший результат точности, чем при 16), активирующая функция –RELU. На выходе - сигмоидная функция – вероятность принадлежности к классу.

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

В качестве оптимизатора используется RMSProp, в качестве функции потерь – бинарная энтропия (обычно используется для бинарной классификации).

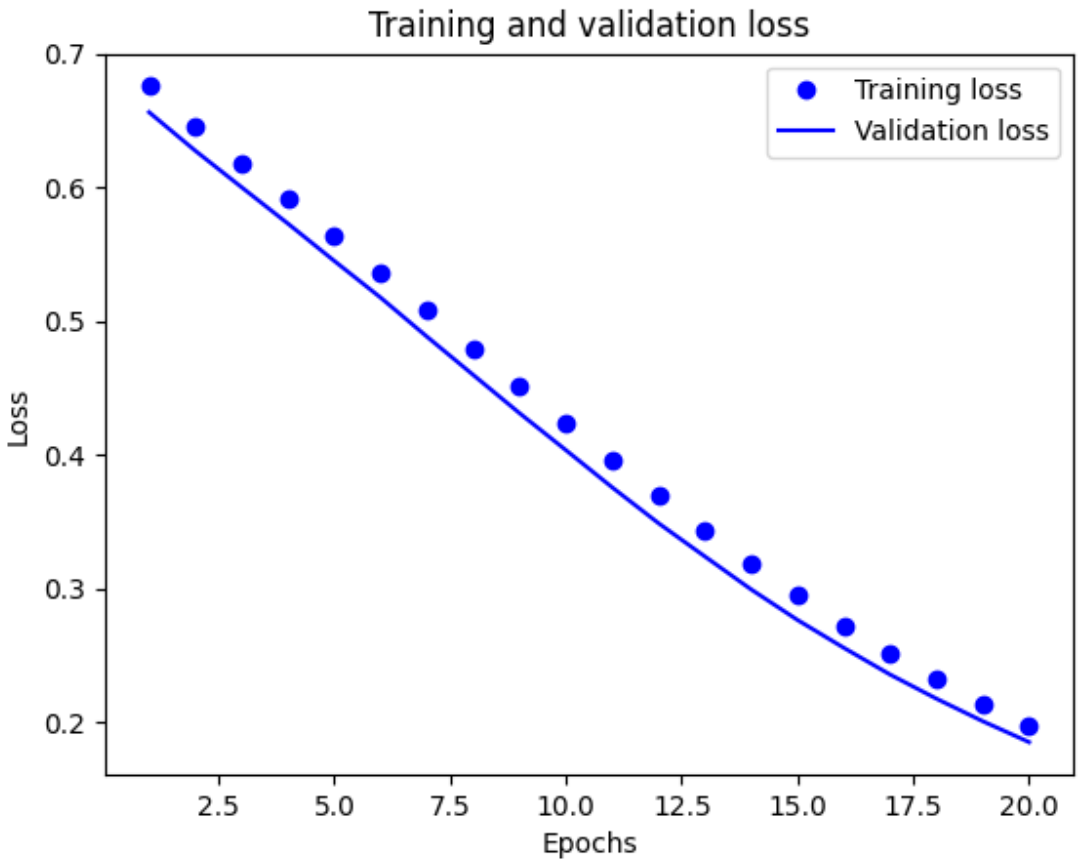
```
data_val = train_data[:len(train_data) * 2 // 5]
data_train = train_data[len(train_data) * 2 // 5:]
label_val = train_label[:len(train_label) * 2 // 5]
label_train = train_label[len(train_label) * 2 // 5:]
```

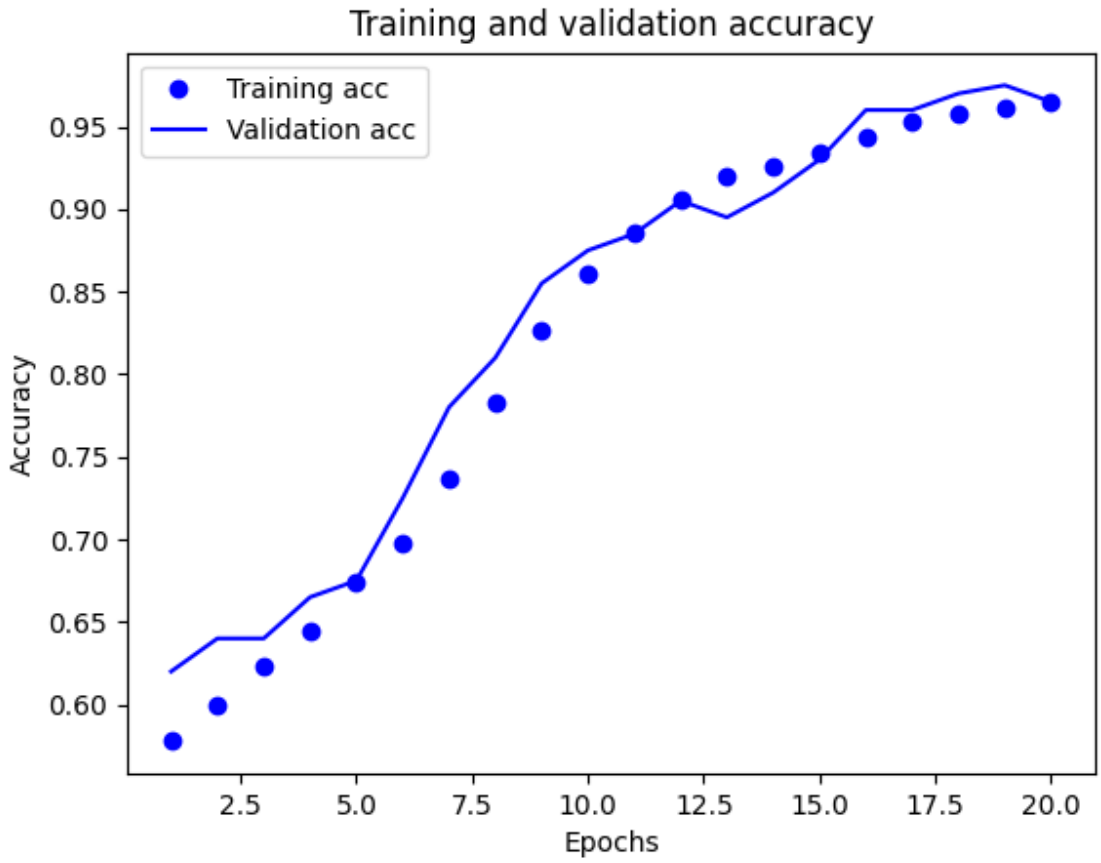
Для валидации отведено 2/5 тренировочных данных.

```
H = model.fit(data_train, label_train, epochs=20, batch_size=500,
validation_data=(data_val, label_val), verbose=False)
```

На этих данных модель обучается 20 эпох, для ускорения обучения размер батча установлен в 500.

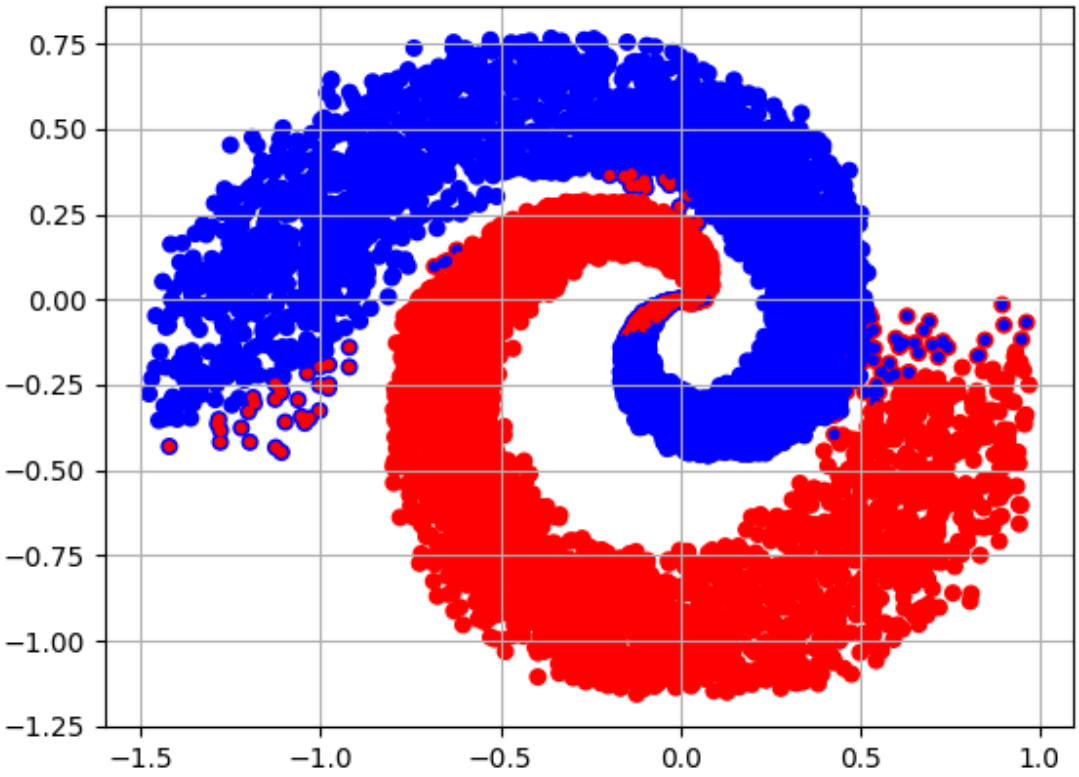
Результаты:






```
model.evaluate(test_data, test_label)
```

Исходя из этой функции, точность данных колеблется от 85 до 90%.



Видно, что на не очень большом наборе данных нейросеть ошибается на пограничных (где ветви функций почти касаются друг друга) и новых (например как в «началах» ветвей) данных.