Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика, искусственный интеллект и системы управления» (ИУ)

КАФЕДРА Системы обработки информации и управления» (ИУ5)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Конструирование нейронной сети для определения вероятности выживания человека при кораблекрушении с применением технологии глубокого обучения

Студент ИУ5-33М (Группа) _____ (Подпись, дата) Фэн Кэцзя (И.О.Фамилия)

Руководитель _____ (Подпись, дата) Ю. Е. Гапанюк (И.О.Фамилия)

Консультант _____ (Подпись, дата) _____ (И.О.Фамилия)

2022 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ5
(Индекс)
_____ _**В.И. Терехов**
(И.О.Фамилия)
« _____ » _____ 20 22 г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме Конструирование нейронной сети для определения вероятности выживания человека при кораблекрушении с применением технологии глубокого обучения

_____

Студент группы ИУ5-33М

_____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

_____

Источник тематики (кафедра, предприятие, НИР) _____

График выполнения НИР: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Техническое задание Конструирование нейронной сети для определения вероятности выживания человека при кораблекрушении с применением технологии глубокого обучения

_____
_____
_____
_____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _____ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____
_____
_____
_____

Дата выдачи задания « ___ » _____ 2022г.

Руководитель НИР _____ Ю. Е. Гапанюк
(Подпись, дата) (И.О.Фамилия)

Студент _____ Фэн Кэцзя
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Содержание

1. Знакомство с миссией

Задача основана на данных о кораблекрушении kaggle «Титаник», после обработки данных строится нейросеть, позволяющая прогнозировать выживаемость экипажа при кораблекрушении в заданных условиях.

2. последовательность задач

2.1 загрузка набора данных

Сюда импортируются необходимые библиотеки и загружается набор данных

Code:

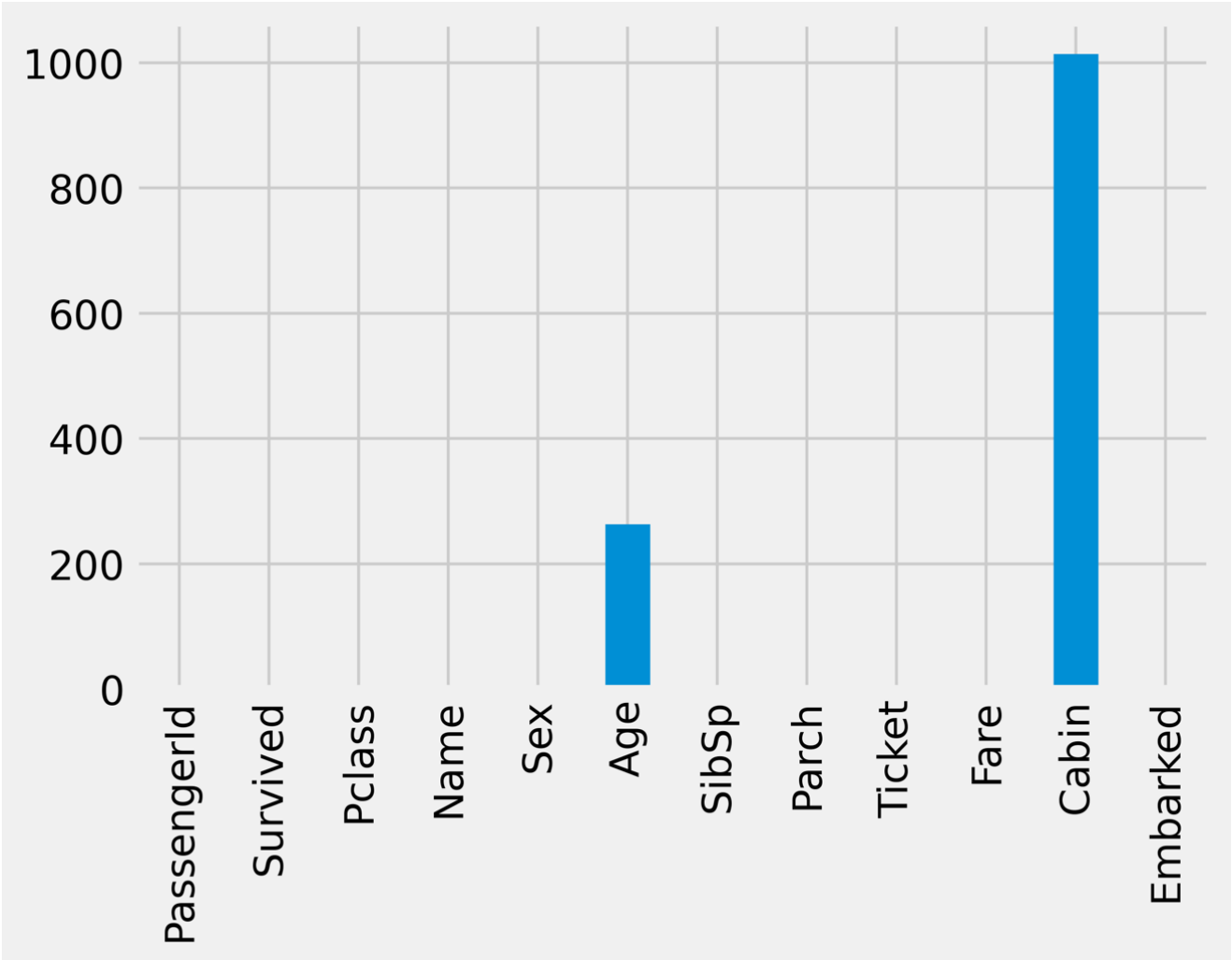
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.preprocessing._encoders import OrdinalEncoder

plt.style.use('fivethirtyeight')
train_file = '/Users/fkj/Desktop/titanic/train.csv'
test_file = '/Users/fkj/Desktop/titanic/tested.csv'

df = pd.read_csv(train_file)
df = df.append(pd.read_csv(test_file))

df.head()
```

2.2 Обработка наборов данных

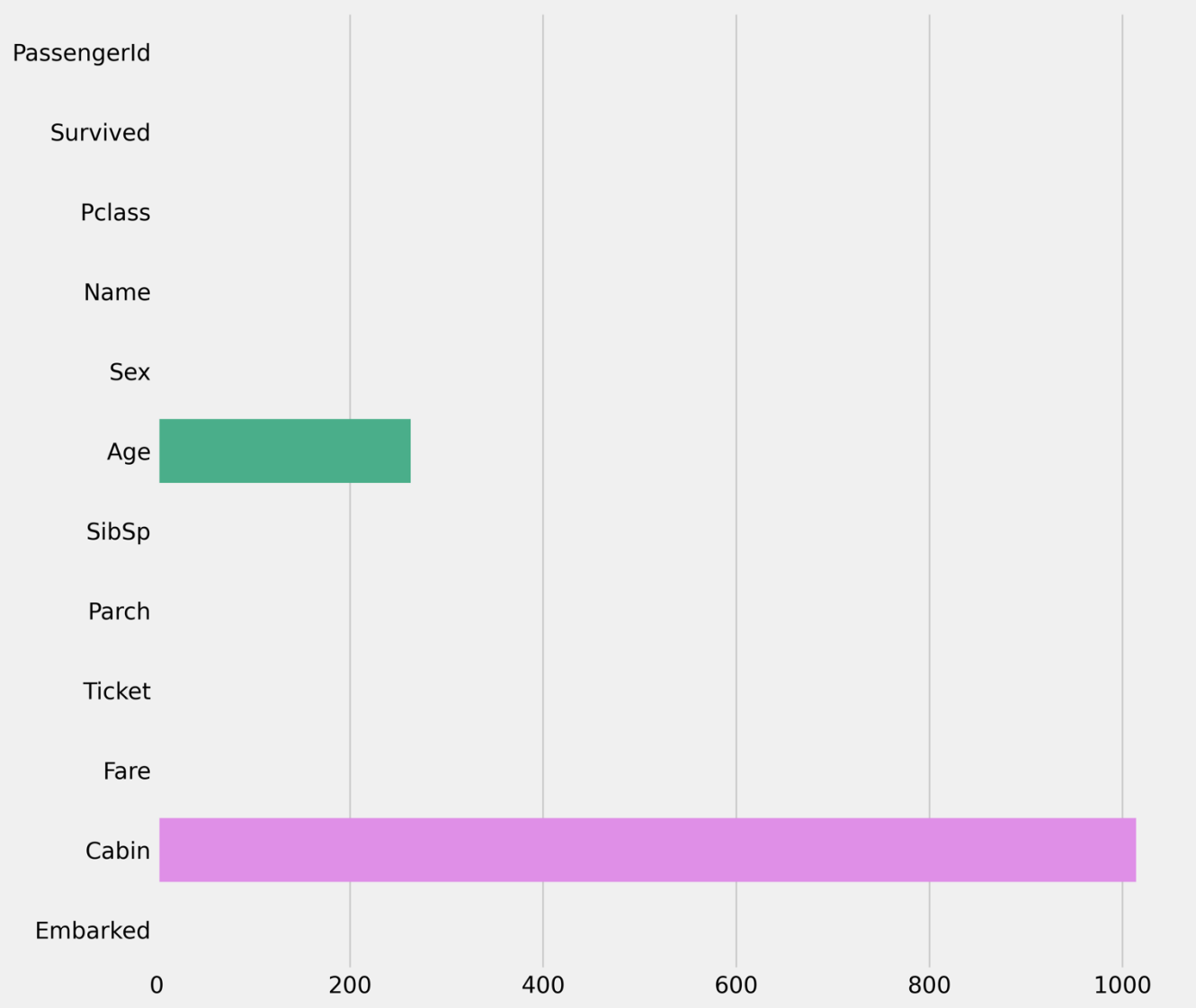
Сначала проверьте набор данных, найдите атрибуты с большим количеством пустых данных и удалите неважные атрибуты из набора данных. Наконец, преобразуйте нечисловые данные в числовые данные.

Code:

```
df.isnull().sum().plot(kind='bar')
plt.savefig('/Users/fkj/Desktop/titanic/0.png', dpi = 400, bbox_inches='tight')
plt.show()
```

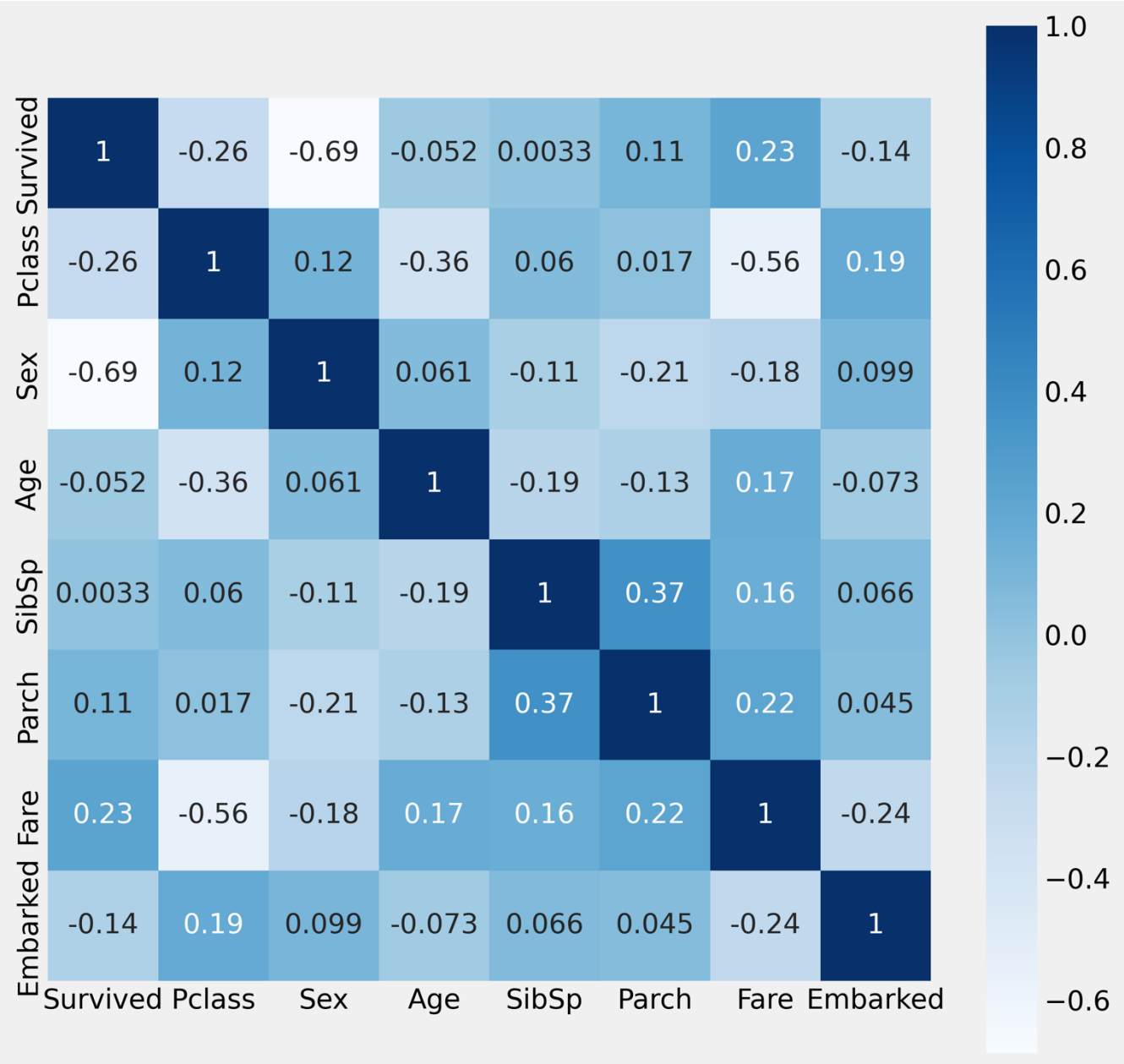



```
s = df.isnull().sum()
fig = sns.barplot(y=s.index,x=s.values,orient='h')
scatter_fig = fig.get_figure()
scatter_fig.savefig('/Users/fkj/Desktop/titanic/1.png', dpi = 400, bbox_inches='tight')
```



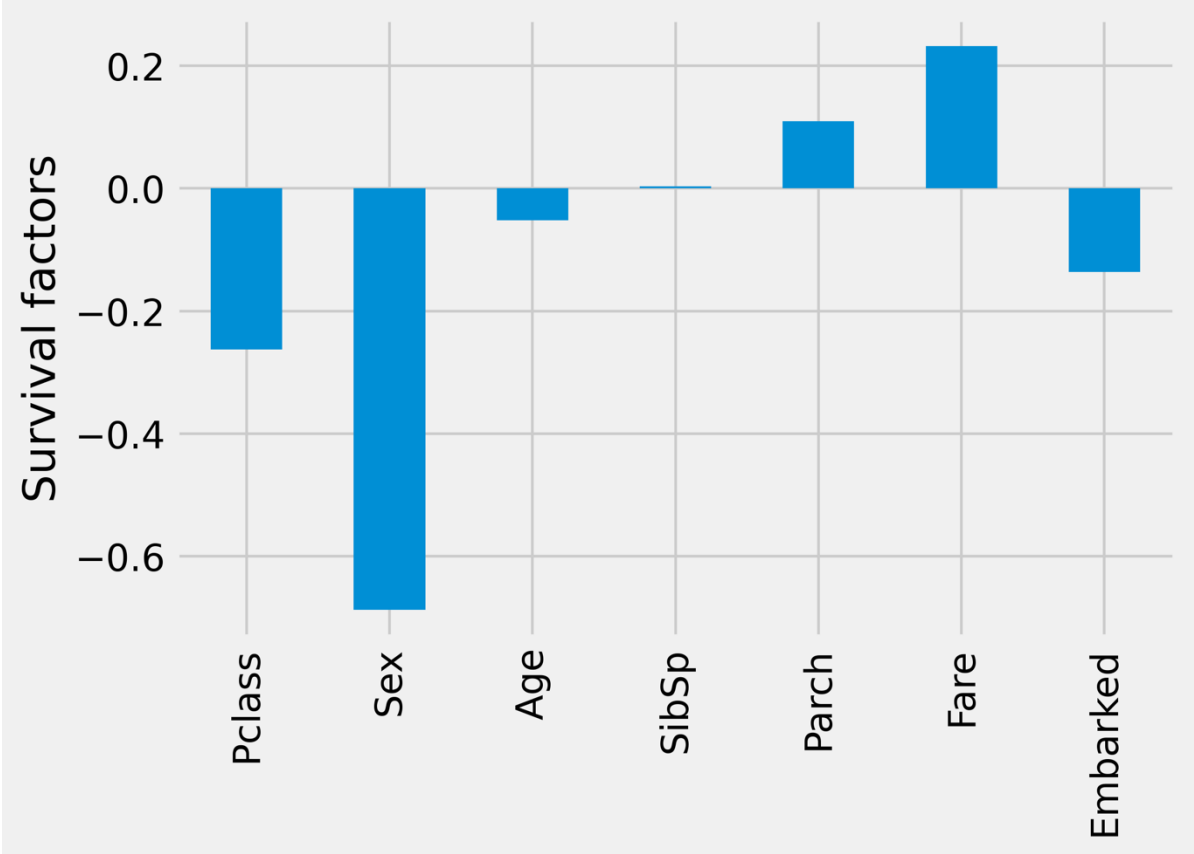

```
df_processed=df.copy().drop(['PassengerId','Name','Ticket','Cabin'],axis=1)
df_processed = df_processed.dropna(subset=['Embarked'])
encoder = OrdinalEncoder()
df_processed.Sex = encoder.fit_transform(df_processed.Sex.to_numpy().reshape(-1,1))
encoder = OrdinalEncoder()
df_processed.Embarked = encoder.fit_transform(df_processed.Embarked.to_numpy().reshape(-1,1))
df_processed.info()

mean_age = df_processed.Age.mean()
df_processed.Age.fillna(mean_age,inplace=True)
df_processed.isnull().sum()
mean_Fare = df_processed.Fare.mean()
df_processed.Fare.fillna(mean_Fare,inplace=True)
```

```
df_processed.info()
df_coor=df_processed.corr()
plt.subplots(figsize=(9,9),facecolor='w')
fig=sns.heatmap(df_coor,annot=True, vmax=1, square=True, cmap="Blues", fmt='.2g')
scatter_fig = fig.get_figure()
scatter_fig.savefig('/Users/fkj/Desktop/titanic/2.png', dpi = 400, bbox_inches='tight')
```




```
bar_plt = df_coor.pop('Survived')
bar_plt.pop('Survived')
bar_plt.plot(kind='bar')
```

```
plt.ylabel("Survival factors")
plt.savefig('/Users/fkj/Desktop/titanic/3.png', dpi = 400, bbox_inches='tight')
plt.show()
```



2.3 Определение модели и загрузка данных

Затем мы определили модель нейронной сети.Эта задача представляет собой задачу классификации с 7 входами и одним выходом, поэтому мы выбрали полносвязную нейронную сеть. После многих попыток было установлено, что существует два скрытых слоя, и в качестве функции активации используется функция relu, а в окончательном выводе используется сигмовидная функция для суждения о двоичной классификации. И используйте модуль netron для визуализации нейронной сети.

Code:

```
from sklearn.model_selection import train_test_split

y = df_processed.pop('Survived')
x = df_processed
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y,test_size=0.2, random_state=7) from sklearn.preprocessing import StandardScaler
# Scling Features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)

from torch import nn

class Network(nn.Module):
    def __init__(self):
        super().__init__()

        self.hidden = nn.Linear(7, 32)
        self.hidden2 = nn.Linear(32, 16)
        self.output = nn.Linear(16, 1)

        self.sigmoid = nn.Sigmoid()
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.hidden(x)
        x = self.relu(x)
        x = self.hidden2(x)
        x = self.relu(x)
        x = self.output(x)
        x = self.sigmoid(x)

        return x

model = Network()

import torch

optimizer = torch.optim.SGD(model.parameters(),lr = 0.1)
```

```
loss_func = torch.nn.MSELoss()

from torch.autograd import Variable

x_tra = Variable(torch.from_numpy(X_train))
x_tra = x_tra.float()
y_tra = np.array(y_train,dtype=np.float64)
y_tra = Variable(torch.from_numpy(y_tra))
y_tra = y_tra.float()

x_val = Variable(torch.from_numpy(X_test))
x_val = x_val.float()
y_val = np.array(y_test,dtype=np.float64)
y_val = Variable(torch.from_numpy(y_val))
y_val = y_val.float()

from torch.utils.data import TensorDataset, DataLoader

batch_size = 100

train_dataset = TensorDataset(x_tra, y_tra)
test_dataset = TensorDataset(x_val, y_val)
train_dataloader = DataLoader(train_dataset, batch_size=batch_size,
shuffle=True)
test_dataloader = DataLoader(test_dataset, batch_size=batch_size, shuffle=True)

train_dataloader

import netron
data_onnx = torch.rand(7)
onnx_path = "/Users/fkj/Desktop/titanic/model.onnx"
torch.onnx.export(model, data_onnx, onnx_path)
netron.start(onnx_path)
```

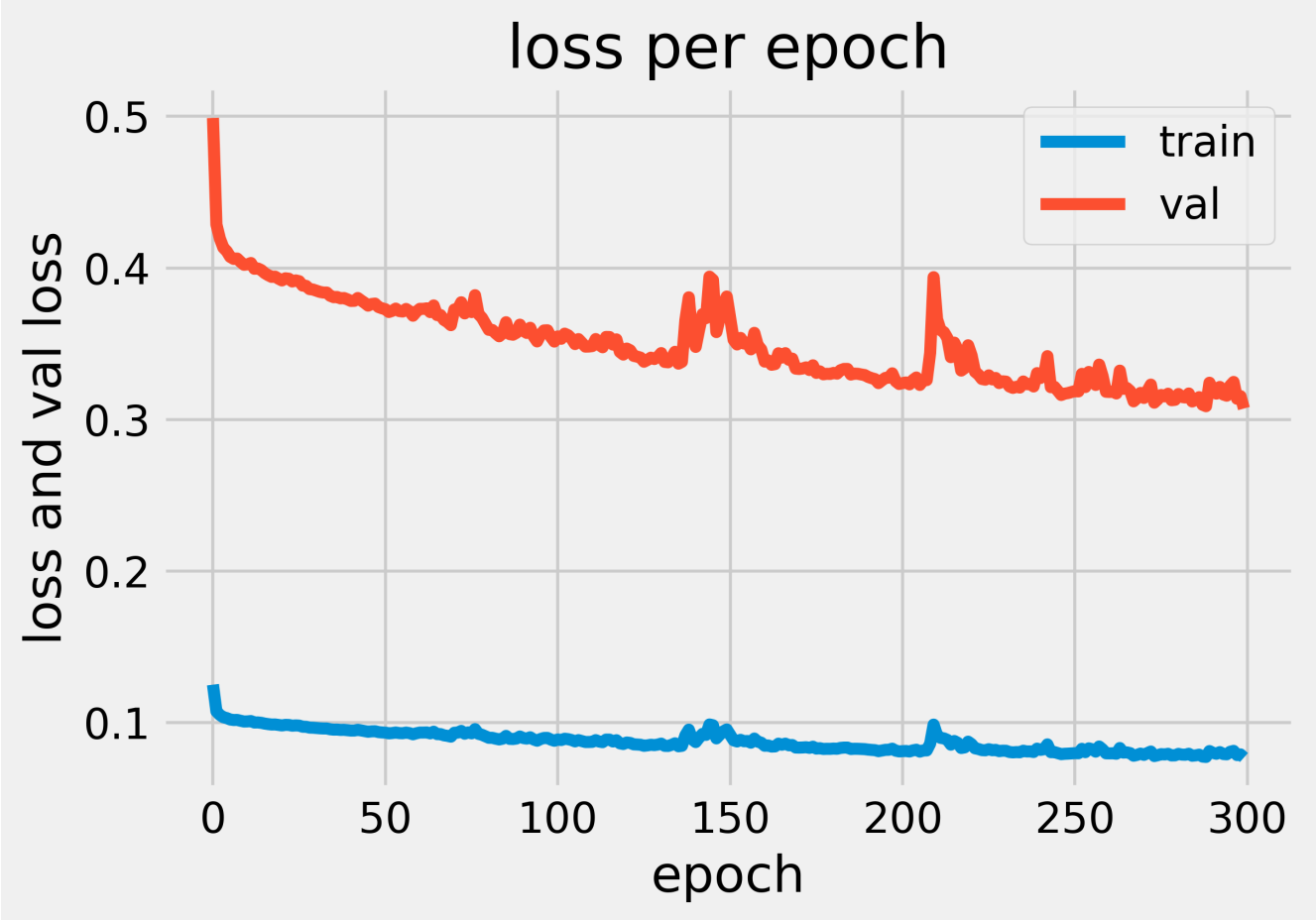
2.4 обучение модели

Далее модель обучается, и после подбора соответствующих гиперпараметров получается приемлемая точность.

Code:

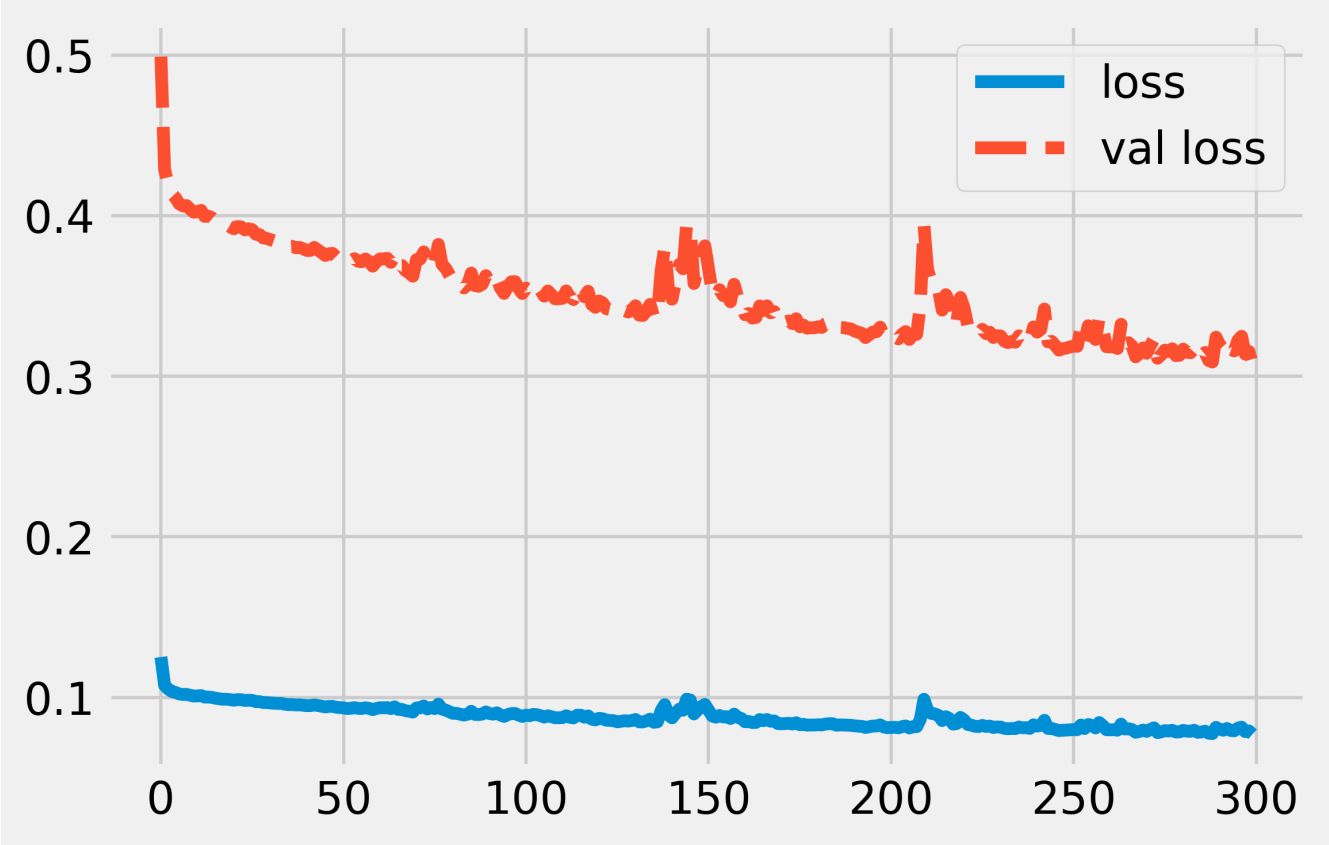
```
from tqdm.auto import tqdm
EPOCHS = 300
REDRAW_EVERY = 5
loss_train_x = []
loss_train_y = []
val_loss_train_x = []
val_loss_train_y = []
steps_per_epoch = len(train_dataset)
steps_per_epoch_val = len(test_dataset)
for epoch in range(EPOCHS):
    running_loss = 0.0
    model.train()
    for i, batch in enumerate(train_dataset, 0):
        inputs, labels = batch

        optimizer.zero_grad()

        outputs = model(inputs)
        loss = loss_func(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss / steps_per_epoch:.3f}')
    loss_train_x.append(epoch)
    loss_train_y.append(running_loss / steps_per_epoch)
    print(f'[{epoch + 1}, {i + 1:5d}] val loss: {running_loss /
steps_per_epoch_val:.3f}')
```

```
    val_loss_train_x.append(epoch)
    val_loss_train_y.append(running_loss / steps_per_epoch_val)
print('Обучение закончено')
plt.plot(loss_train_x, loss_train_y, label='train')
plt.plot(val_loss_train_x, val_loss_train_y, label='val')
plt.grid(which='minor', c='lightgrey')
plt.xlabel("epoch")
plt.ylabel("loss and val loss")
plt.title("loss per epoch")
plt.legend(loc='best')
plt.savefig('/Users/fkj/Desktop/titanic/4.png', dpi = 400, bbox_inches='tight')
plt.show()
```




```
some_dict = {}
some_dict['loss']=loss_train_y
some_dict['val loss']=val_loss_train_y
loss_train = pd.DataFrame(some_dict)
fig = sns.lineplot(data=loss_train)
```

```
scatter_fig = fig.get_figure()
scatter_fig.savefig('/Users/fkj/Desktop/titanic/5.png', dpi = 400,
bbox_inches='tight')
```



3. Прогнозирование модели и отображение точности

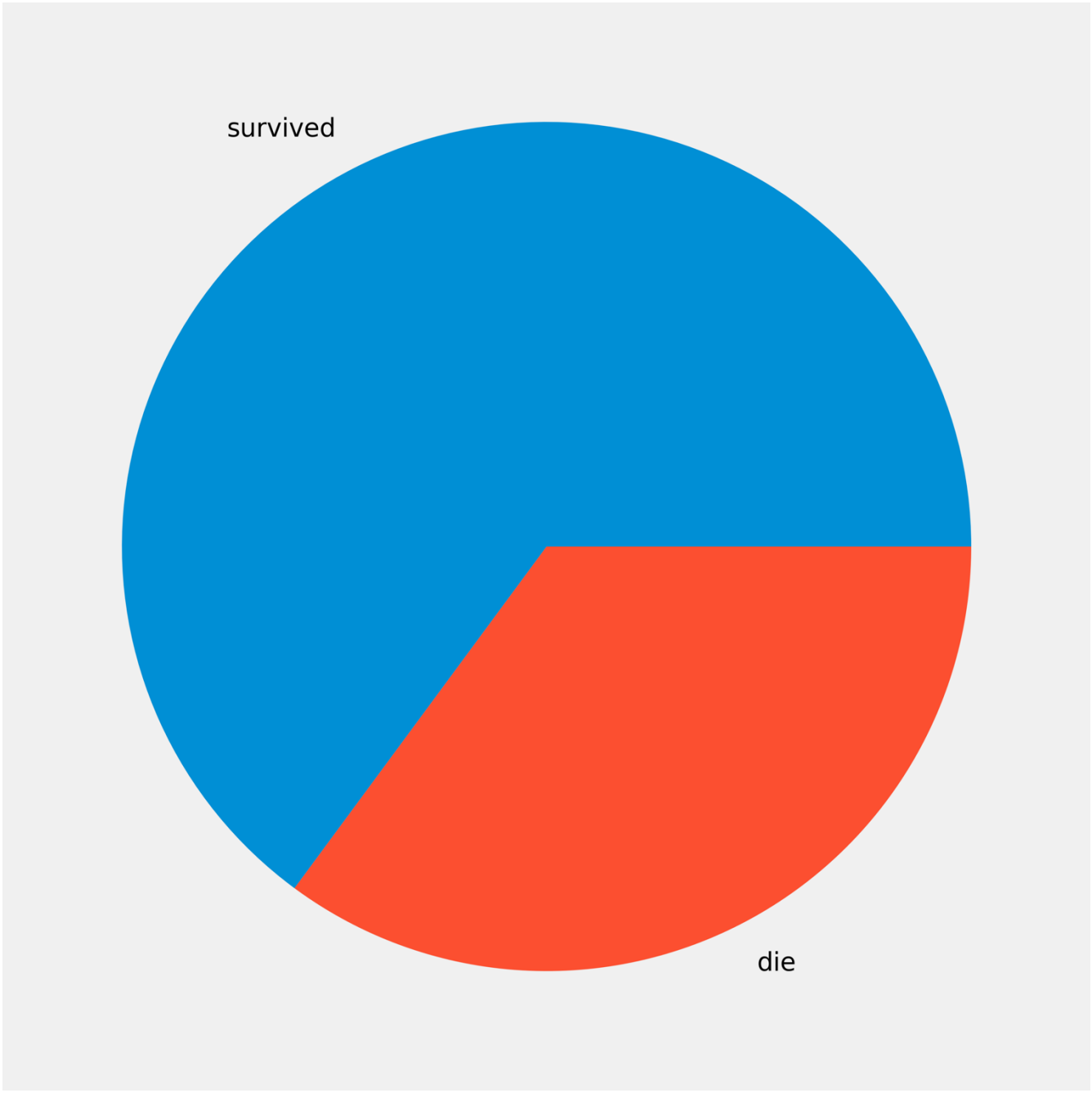
Я выбрал клиента номер 5 для предсказания. Прогнозируемый коэффициент выживаемости, выведенный моделью, составлял 0,0019, и фактические пассажиры действительно погибли в кораблекрушении.

На следующих круговых диаграммах и гистограммах показаны сходства и различия между предсказаниями модели и фактическими значениями данных. Видно, что предсказания модели очень близки к реальным данным.

Code:

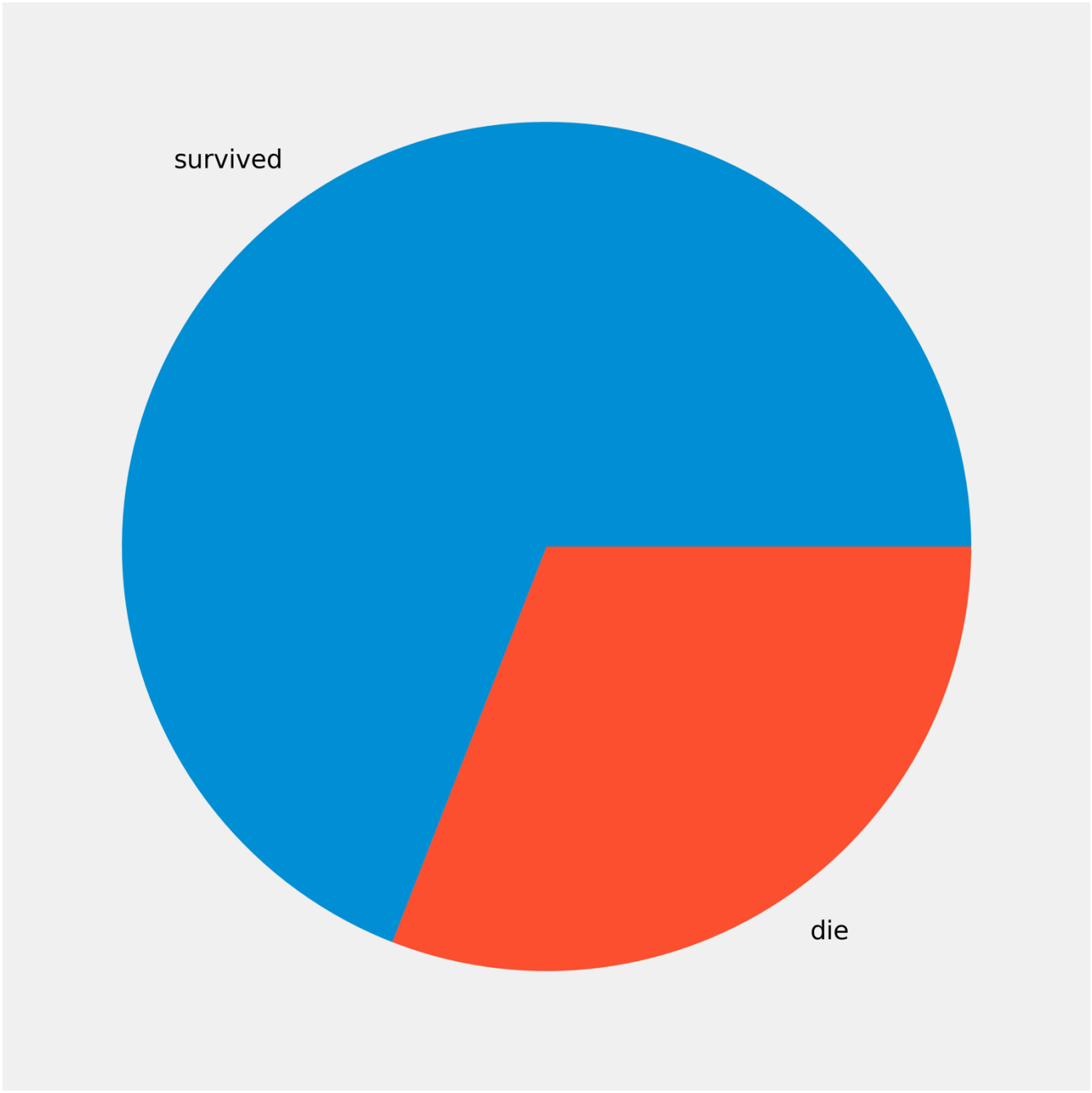
```
testman = X_train[5]
testman = Variable(torch.from_numpy(testman))
testman = testman.float()
pred = model(testman)
print(pred)
y_train[5]
y_test_pred = []
for i, batch in enumerate(test_dataset, 0):
        inputs, labels = batch

        optimizer.zero_grad()

        outputs = model(inputs)
        y_test_pred += outputs
        loss = loss_func(outputs, labels)
        loss.backward()
        optimizer.step()

y_test_pred
def step(x=''):
    if x >0.5:
        return 1
    else:
        return 0
y_test_pred_np = []
for i in y_test_pred:
    a = i.detach().numpy()
    b = step(a)
    #print(b)
    y_test_pred_np.append(b)

print(y_test_pred_np)
y_test_true = []
for i in y_test:
    y_test_true.append(i)
```

```
y_test_true
yp = pd.Series(y_test_pred_np).value_counts()
yt = pd.Series(y_test_true).value_counts()
yt
plt.rcParams['figure.figsize']=[10,10]
labels =[ 'survived','die']
plt.pie(yt,labels=labels)
plt.savefig('/Users/fkj/Desktop/titanic/6.png', dpi = 400, bbox_inches='tight')
plt.show()
```



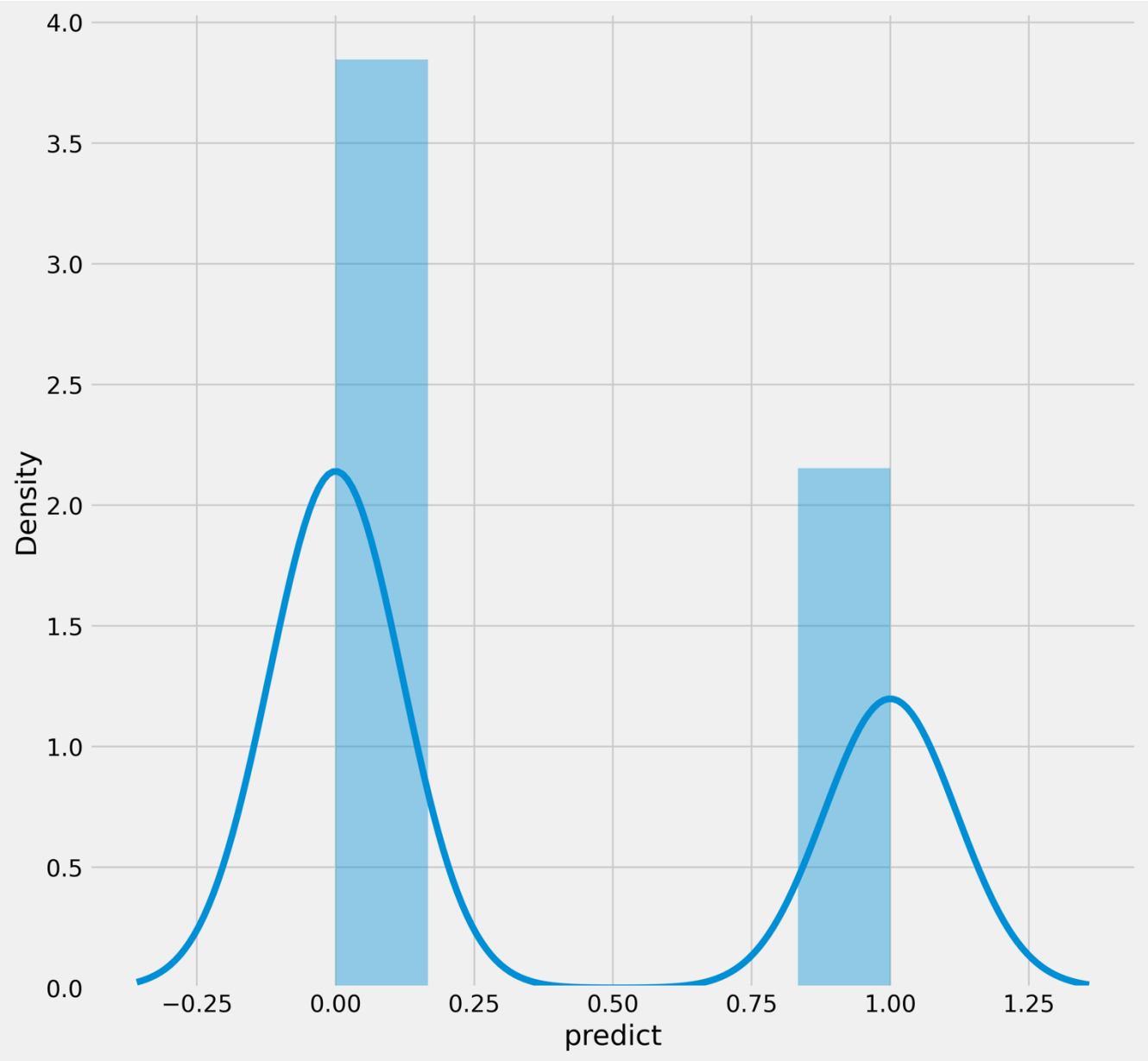

```
plt.rcParams['figure.figsize']=[10,10]
```

```
labels =[ 'survived','die']
plt.pie(yp,labels=labels)
plt.savefig('/Users/fkj/Desktop/titanic/7.png', dpi = 400, bbox_inches='tight')
plt.show()
```

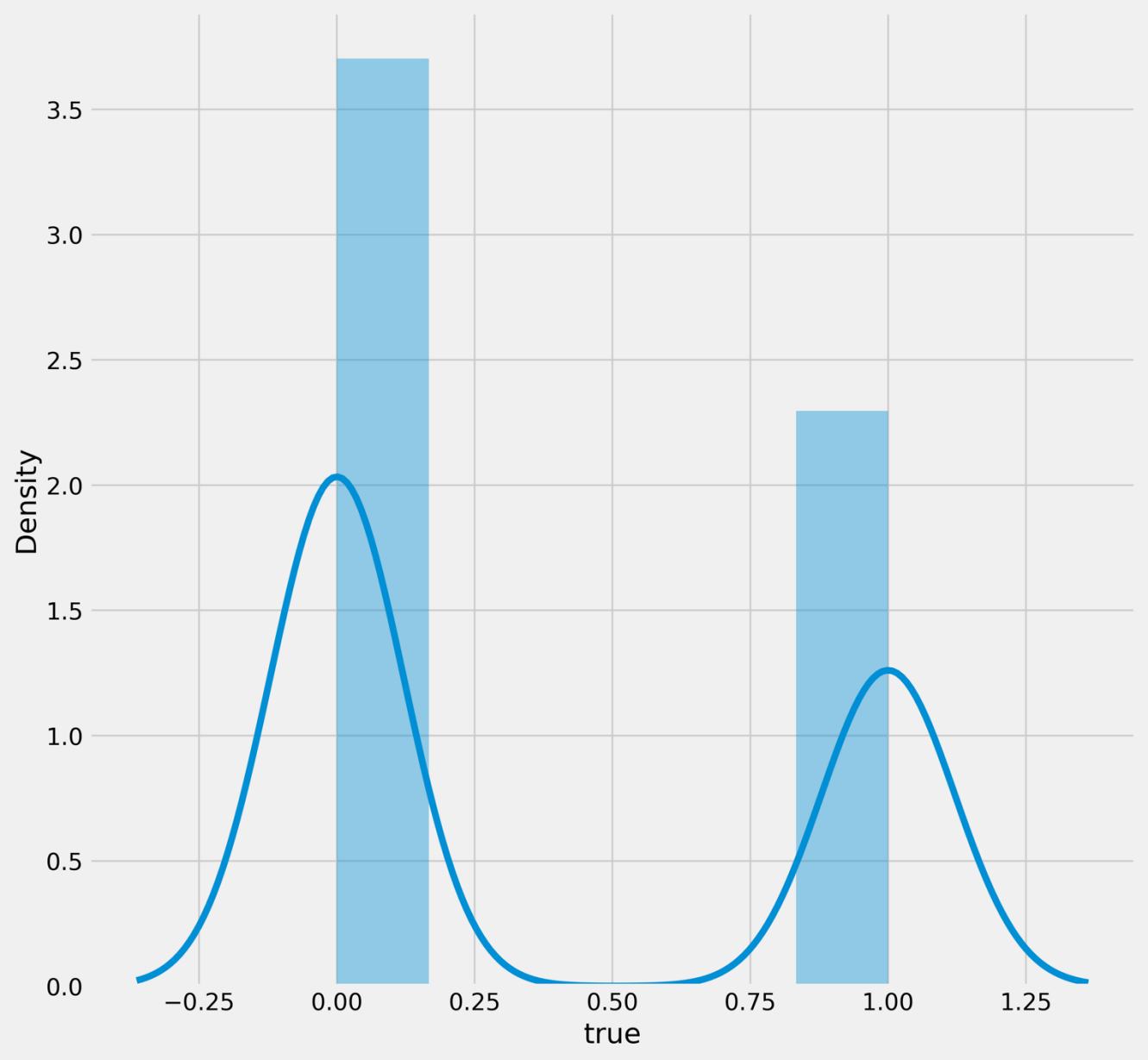



```
y_train_pred = []

for i, batch in enumerate(train_dataset, 0):
    inputs, labels = batch

    optimizer.zero_grad()
```

```
            outputs = model(inputs)
            y_train_pred += outputs
            loss = loss_func(outputs, labels)
            loss.backward()
            optimizer.step()

        y_train_pred

        y_train_pred_np = []

        for i in y_train_pred:
            a = i.detach().numpy()
            b = step(a)
            #print(b)
            y_train_pred_np.append(b)

        print(y_train_pred_np)

        y_train_true = []

        for i in y_train:
        y_train_true.append(i)

        y_train_true

        ypdf = pd.DataFrame(y_train_pred_np,columns=['survived'])
        ytdf = pd.DataFrame(y_train_true,columns=['survived'])

        plt.xlabel('predict')
        fig = sns.distplot(ypdf)
        scatter_fig = fig.get_figure()
        scatter_fig.savefig('/Users/fkj/Desktop/titanic/8.png', dpi = 400,
bbox_inches='tight')
```




```
    plt.xlabel('true')
    fig = sns.distplot(ytdf)
    scatter_fig = fig.get_figure()
    scatter_fig.savefig('/Users/fkj/Desktop/titanic/9.png', dpi = 400,
bbox_inches='tight')
```



Вывод

В этой статье подробно описан процесс создания нейронной сети для прогнозирования выживания с использованием набора данных Kaggle Titanic. Сложность этой программы в том, что изменение типов данных и структур часто приводит к ошибкам, что свидетельствует о том, что изменения типов данных в потоке программы должны быть освоены умело.