Лабораторна робота № 10. Моделювання нейронної мережі. Розв’язання класичної задачі «Титанік»

Мета: сформулювати практичні навички розробки та тренування нейронних мереж за допомогою сучасних бібліотек глибокого навчання. Робота спрямована на розвиток вмінь аналізувати та обробляти реальні набори даних, виконувати інженерію ознак, підготовку даних до тренування моделі, а також оптимізацію гіперпараметрів для досягнення найкращої можливої точності прогнозування. Ця робота також має на меті розвинути розуміння процесів, які лежать в основі глибокого навчання, та показати, як ці технології можуть бути застосовані для розв’язання практичних завдань.

Теоретичні відомості

Глибоке навчання є підгалуззю **машинного навчання**, яке використовує нейронні мережі з численними прихованими шарами для моделювання складних абстракцій. Воно знайшло широке застосування у багатьох областях, таких як комп'ютерний зір, обробка природної мови, рекомендаційні системи та інші.

Нейронні мережі імітують роботу людського мозку і складаються з великої кількості вузлів, званих нейронами, які з’єднані між собою зв'язками. Кожен зв’язок має вагу, що регулює вплив одного нейрона на інший. Навчання нейронної мережі полягає у визначенні оптимальних значень цих ваг на основі навчальних даних.

Задача «Титанік» є класичним прикладом бінарної класифікації, де необхідно на основі набору характеристик пасажирів (вік, стать, клас каюти та інші) передбачити, чи вижив пасажир під час катастрофи. Ця задача вимагає від дослідника виконання кількох ключових етапів: попередньої обробки даних, включаючи заповнення пропущених значень та кодування категоріальних ознак; інженерії ознак для створення нових змінних, які можуть покращити модель; тренування моделі, використовуючи нейронні мережі; а також оцінку точності моделі на тестовому наборі даних.

Для ефективного тренування моделі важливо вибрати відповідні гіперпараметри, такі як швидкість навчання, кількість епох, розмір партії та архітектура мережі (кількість шарів і нейронів в кожному шарі). Оптимізація гіперпараметрів може значно покращити продуктивність моделі.

Приклад виконання роботи

Завдання

Титанік – британський пасажирський лайнер компанії White Star Line, який затонув у північній частині Атлантичного океану рано вранці 15 квітня 1912 року після зіткнення з айсбергом під час свого першого рейсу з Саутгемптона до Нью-Йорка з приблизно 2224 пасажирами і членами екіпажу, які перебували на борту, понад 1500 загинули, що робить цю катастрофу однією з найсмертоносніших комерційних морських катастроф мирного часу в сучасній історії. Хоча на борту було близько 2224 пасажирів і членів екіпажу, ми маємо дані про 1300 пасажирів. З цих 1300, близько 900 даних використовуються з навчальною метою, а решта 400 – з тестовою. У цьому завдані вам надано близько 400 тестових даних з відсутнім стовпчиком вижившиx. Вам треба змоделювати модель нейронної мережі, щоб передбачити, чи вижили пасажири у тестових даних, чи ні. Як навчальні, так і тестові дані не є чистими (містять багато пропущених значень). Побудуйте модель з найкращою точністю.

Характеристика даних:

- Survival (виживання): 0 – ні, 1 – так;
- Pclass (номер класу): 1-й – вищий, 2-й – середній, 3-й – нижчий;
- sibsp (кількість братів і сестер / подружжя на борту): Sibling – брат, сестра, зведений брат, зведена сестра, Spouse – чоловік, дружина (коханки та наречені не враховуються);
- parch (кількість батьків / дітей на борту): Parent – мати, батько, Child – донька, син, падчерка, пасинок. Деякі діти подорожували лише з нянею, тому для них parch = 0;
- Ticket (квиток): номер квитка;
- Fare (вартість): пасажирський тариф;
- Cabin (каюта): номер каюти;

- Port of Embarkation (порт посадки): C – Cherbourg, Q – Queenstown, S – Southampton;
- Name (ім’я), Sex (стать), Age (вік).

Контрольне запитання №1

Вкажіть результуюче значення Accuracy.
Відповідь: _____

Контрольне запитання №2

Вкажіть результуюче значення Precision.
Відповідь: _____

Контрольне запитання №3

Вкажіть яке значення Survived приймає PassengerId 900.
Відповідь: _____

Хід роботи

1. Збережемо та імпортуємо набори даних.

	A	B	C	D	E	F	G	H	I	J	K	L
1	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr. Owen H	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, Mrs. John I	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen, Miss. Lain	female	26	0	0	STON/O2. 3101282	7.925		S
5	4	1	1	Futrelle, Mrs. Jacque	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr. William H	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
8	7	0	1	McCarthy, Mr. Timot	male	54	0	0	17463	51.8625	E46	S

Рисунок 9.1. Датасет на Google Drive

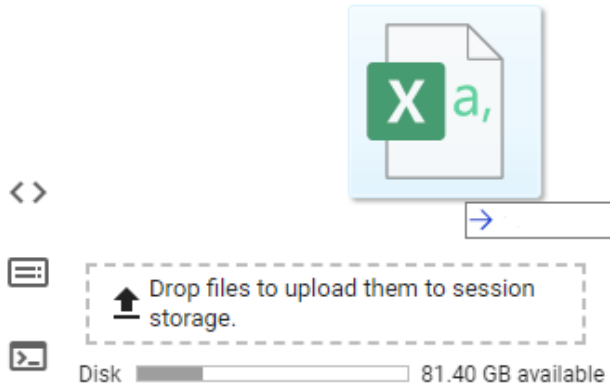

Рисунок 9.2. Завантаження датасету до Google Colab

2. Встановлення необхідних бібліотек та модулів.

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import tensorflow as tf
import keras
from keras.layers import Dense, Dropout, Input
from keras.models import Sequential
```

3. Завантаження наборів даних.

```
train_data = pd.read_csv('/content/train.csv')
test_data = pd.read_csv('/content/test.csv')

train_data.head()
test_data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Рисунок 9.3. Перші 5 рядків навчального набору даних

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Рисунок 9.4. Перші 5 рядків тестового набору даних

Ми бачимо, що загальна кількість стовпців у тестовому наборі даних на один менше, ніж у навчальному. Відсутній стовпець у тестових даних – це Survived (Вижившi) стовпець, який ми повинні передбачити.

```
print("Загальна кількість рядків у навчальному наборі даних ",
train_data.shape[0])
print("Загальна кількість стовпців у навчальному наборі даних ",
train_data.shape[1])
print("Загальна кількість рядків у тестовому наборі даних ",
test_data.shape[0])
print("Загальна кількість стовпців у тестовому наборі даних ",
test_data.shape[1])
```

```
Загальна кількість рядків у навчальному наборі даних  891
Загальна кількість стовпців у навчальному наборі даних  12
Загальна кількість рядків у тестовому наборі даних  418
Загальна кількість стовпців у тестовому наборі даних  11
```

Рисунок 9.5. Загальна кількість рядків та стовпців у тестовому та навчальному наборах даних

4. Візуалізація та аналіз даних

Виконаємо візуалізацію кількості нульових значень в обох наборах даних

```
plt.figure(figsize = (13,5))
plt.bar(train_data.columns, train_data.isna().sum())
plt.xlabel("Назва стовпця")
plt.ylabel("Кількість NULL значень у навчальному наборі даних")
plt.show()
```

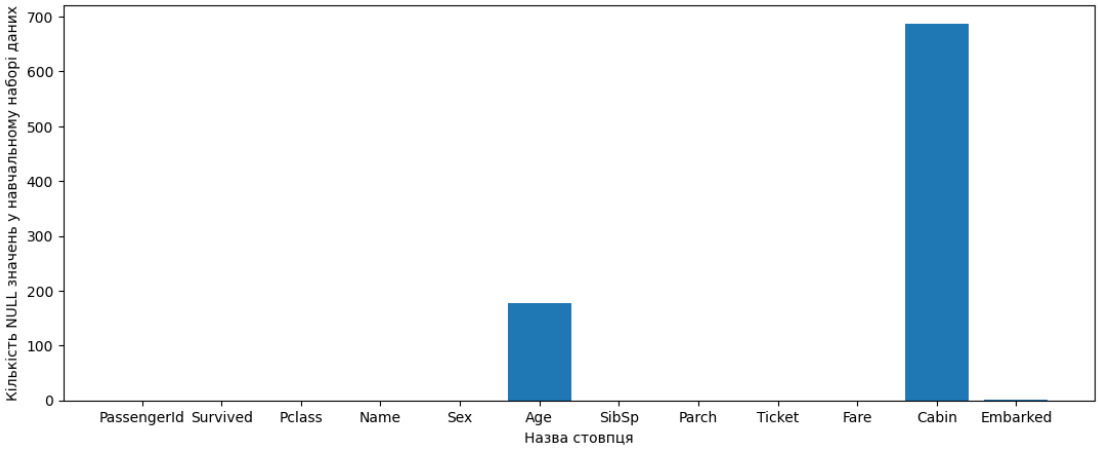

Рисунок 9.6. Графік «Кількість NULL значень у навчальному наборі даних»

```
plt.figure(figsize = (13,5))
plt.bar(test_data.columns, test_data.isnull().sum().values, color = 'red')
plt.xlabel("Назва стовпця")
plt.ylabel("Кількість NULL значень у тестовому наборі даних")
plt.show()
```

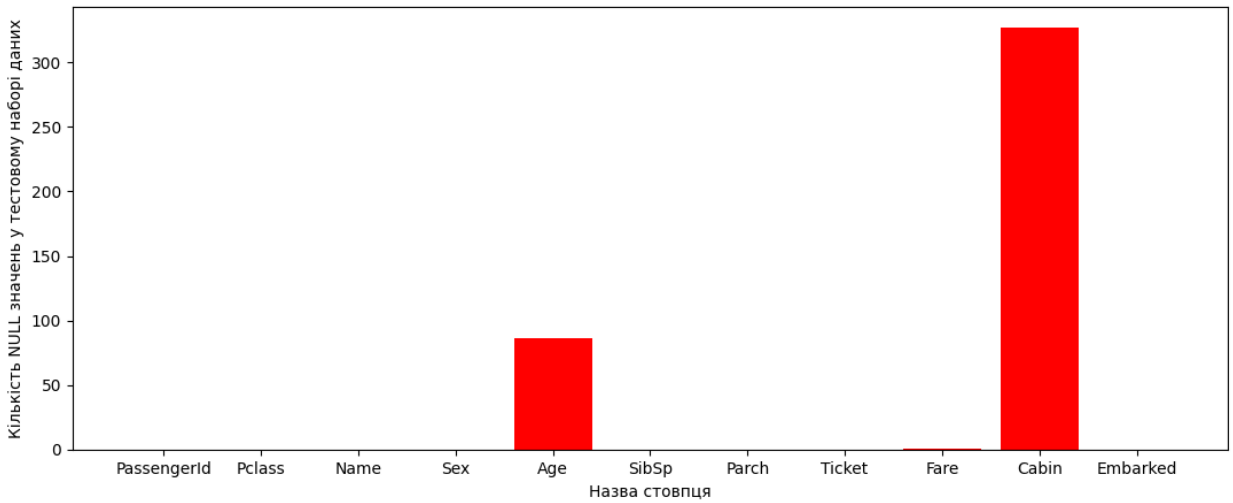

Рисунок 9.7. Графік «Кількість NULL значень у тестовому наборі даних»

Виконаємо візуалізацію кількості пасажирів, що вижили.

```
# Візуалізація кількості пасажирів, що вижили
sns.countplot(x='Survived', data = train_data)
plt.show()
```

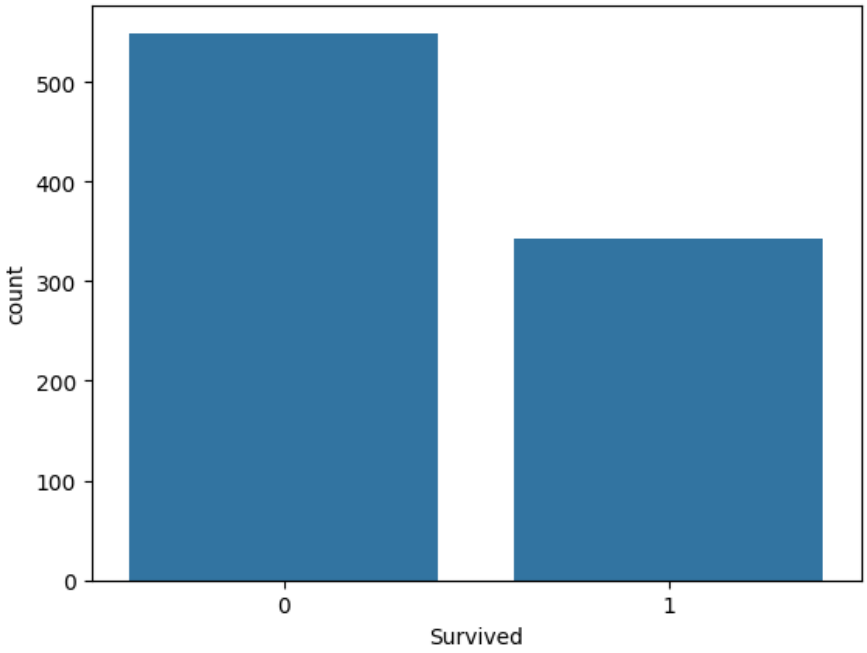

Рисунок 9.8. Графік «Кількість пасажирів, що вижили»

Виконаємо візуалізацію кількості пасажирів з посадкою у різних портах.

```
# Візуалізація кількості пасажирів з різних портів у навчальному наборі даних
sns.countplot(x='Embarked', data = train_data)
plt.show()
```

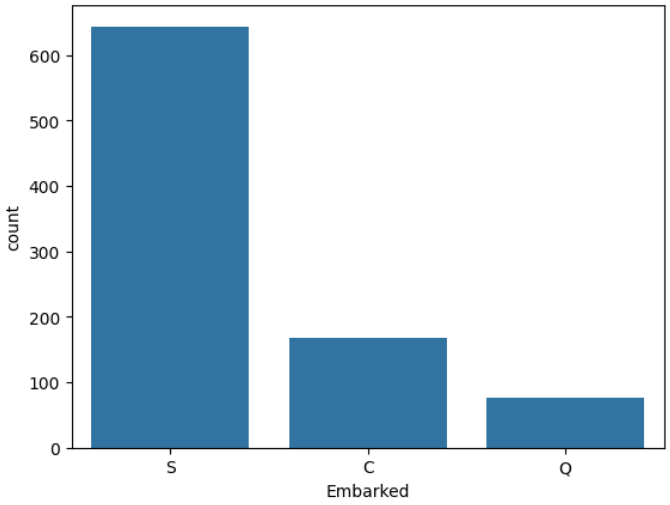

Рисунок 9.9. Графік «Кількість пасажирів з різних портів»

Виконаємо візуалізацію впливу гендеру на рівень виживання.

```
# Візуалізація впливу гендеру на рівень виживання
sns.countplot(x='Survived', hue = 'Sex', data = train_data)
plt.plot()
```

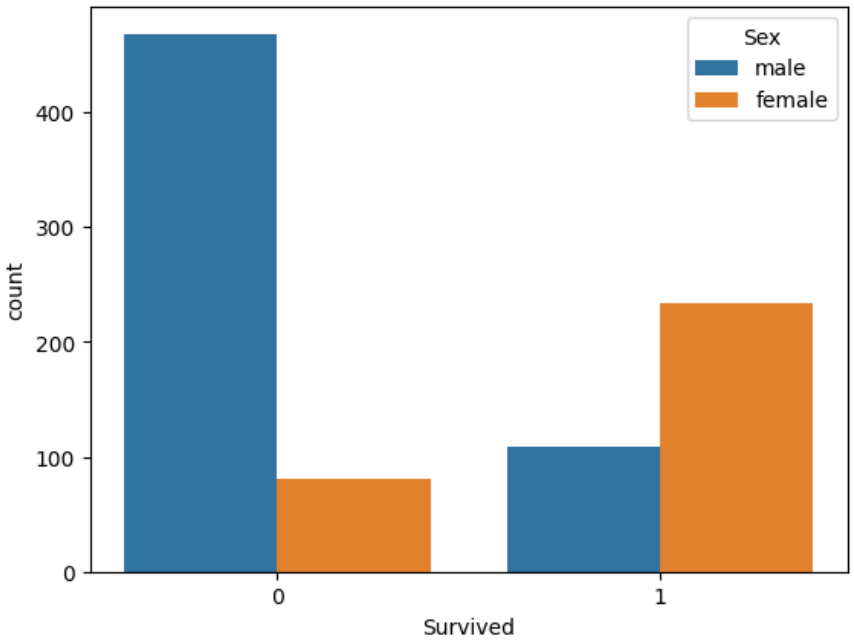

Рисунок 9.10. Графік «Вплив гендеру на виживання»

Виконаємо візуалізацію впливу рівня класу (pclass) на рівень виживання.

```
# Візуалізація впливу рівня класу (pclass) на рівень виживання
sns.countplot(x="Survived", hue = 'Pclass', data = train_data)
plt.show()
```

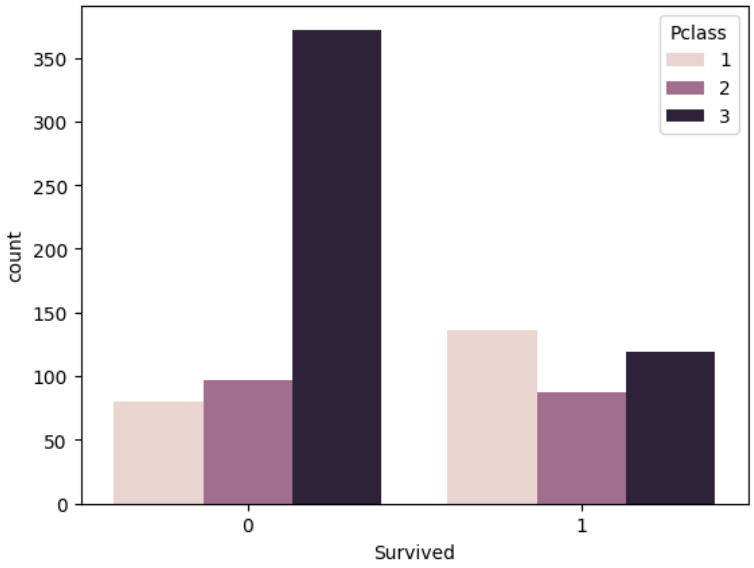

Рисунок 9.11. Графік «Вплив рівня класу (pclass) на рівень виживання»

Виконаємо візуалізацію впливу порту посадки на рівень виживання

```
# Візуалізація впливу порту посадки на рівень виживання
sns.countplot(x='Survived', hue = 'Embarked', data = train_data)
plt.show()
```

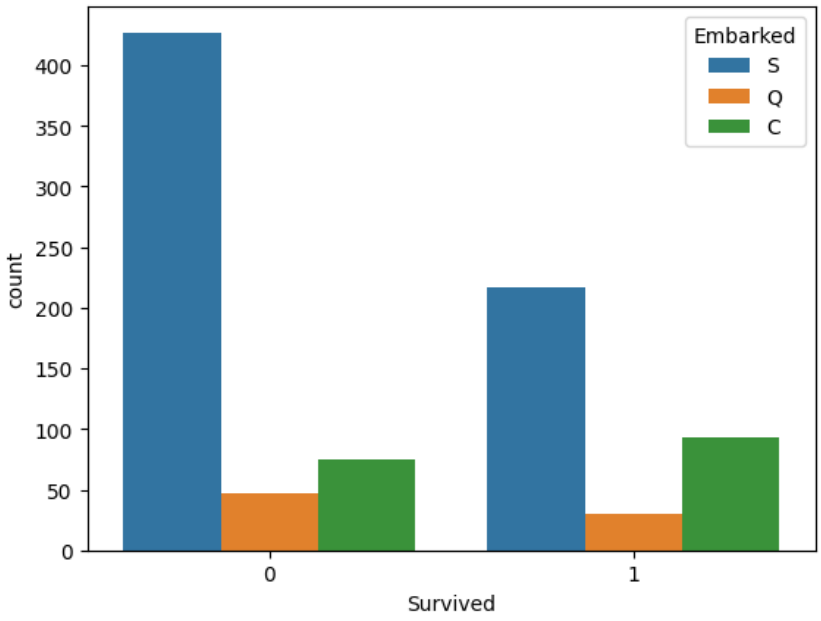

Рисунок 9.12. Графік «Вплив порту посадки на рівень виживання»

```
sns.boxplot(x='Fare', data = train_data)
plt.show()
# Результати свідчать про те, що було дуже мало людей, які заплатили
більше 100$ за квиток
```

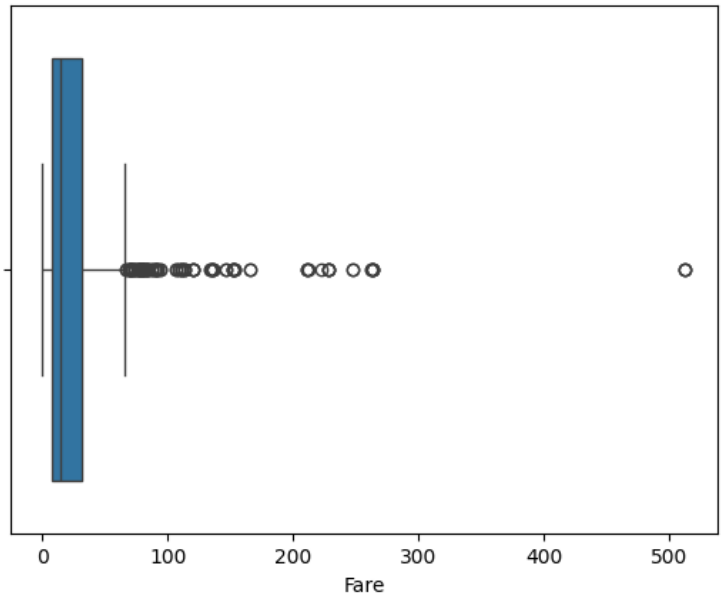

Рисунок 9.13. Графік викидів «Fare»

```
sns.boxplot(x='Age', data = train_data)
plt.show()
# Результати свідчать про те, що в навчальних даних було дуже мало людей,
старших за 65 років
```

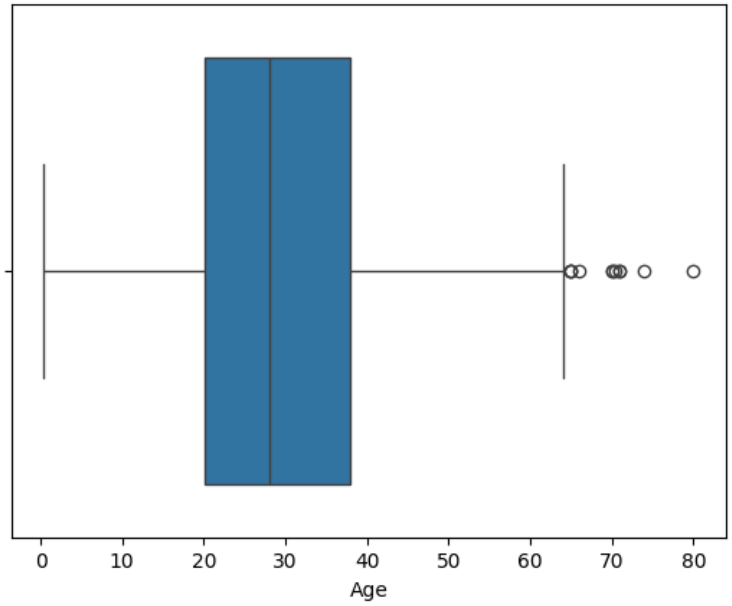

Рисунок 9.14. Графік викидів «Age»

```
interval = 10
value_for_bin = np.ceil((train_data.Age.max() - train_data.Age.min()) /
interval).astype(int)

plt.hist(train_data.Age, bins = value_for_bin)
plt.xlabel("Age")
plt.ylabel("Number")
plt.show()
# Результати свідчать про те, що більшість пасажирів були у віці від 20 до
40 років
```

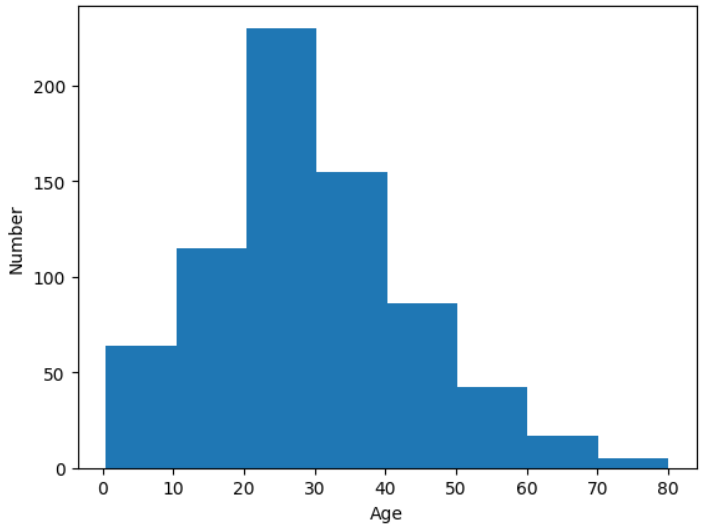

Рисунок 9.15. Графік «Віковий діапазон пасажирів»

```
plt.figure(figsize = (10,4))
plt.hist(train_data.Fare, bins = 10, color = 'lime')
plt.xlabel("Fare")
plt.ylabel("Number")
plt.show()
# Результати свідчать, що близько 700 людей заплатили від 0 до 50$
```

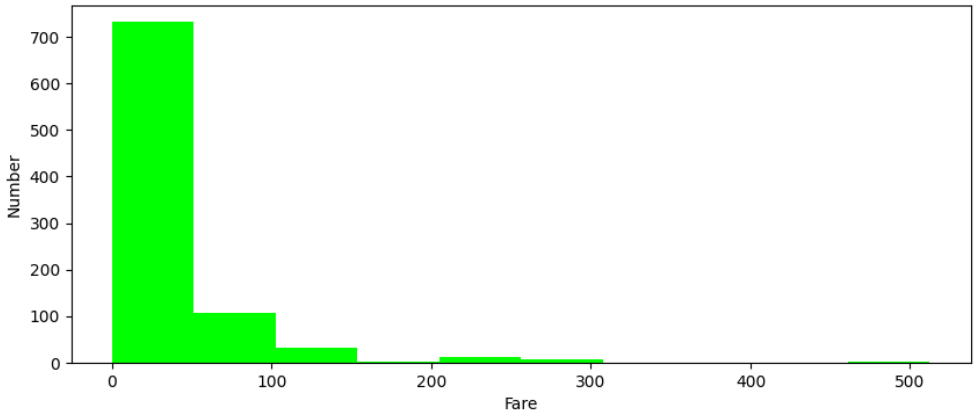

Рисунок 9.16. Графік «Витрати на квиток»

```
grid = sns.FacetGrid(train_data, col='Survived', row='Pclass', height=2.2,
aspect=1.6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend()
plt.show()
```

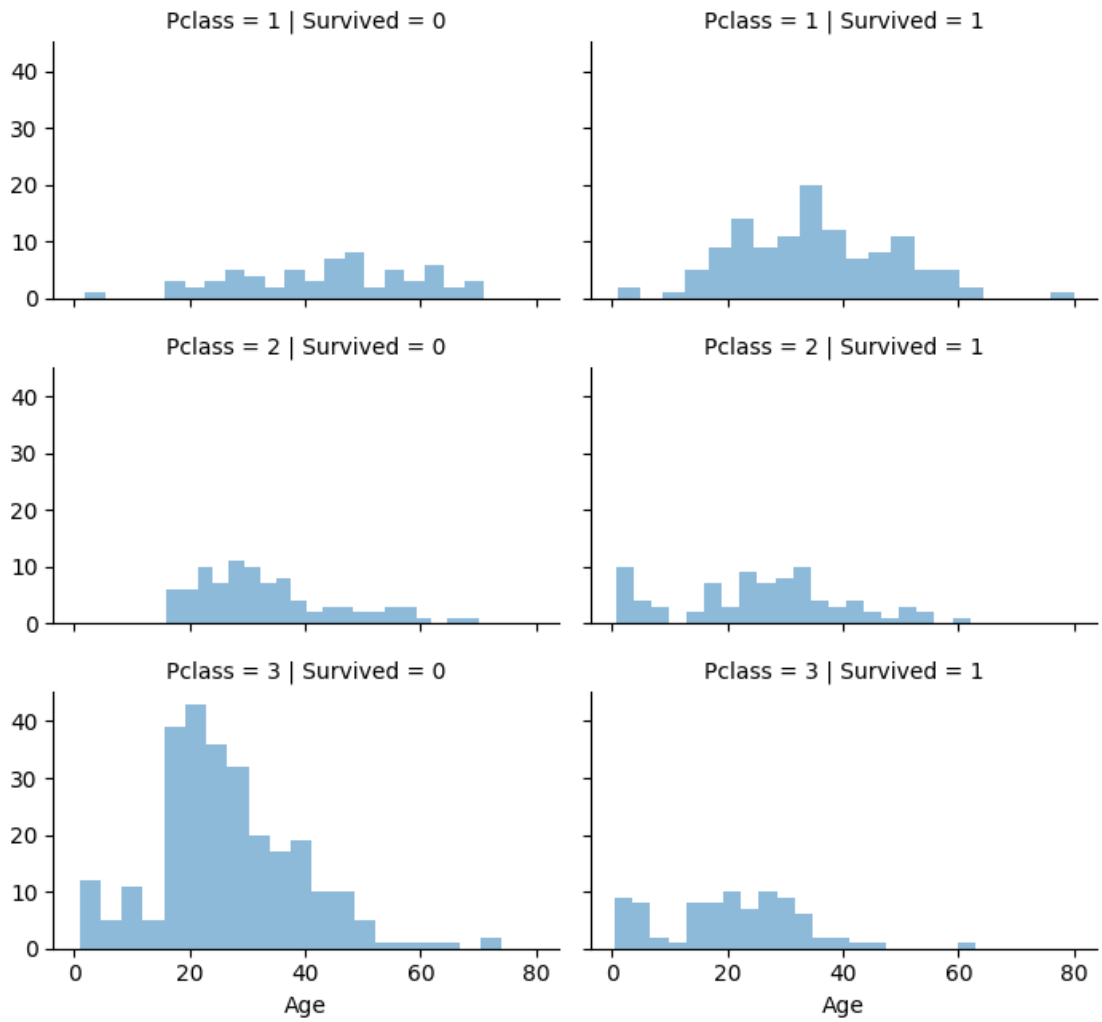

Рисунок 9.17. Графіки співвідношення класу та виживання

```
grid = sns.FacetGrid(train_data, row='Embarked', col='Survived',
height=2.2, aspect=1.6)
grid.map(sns.barplot, 'Sex', 'Fare', alpha=.5, ci=None)
grid.add_legend()
plt.show()
```

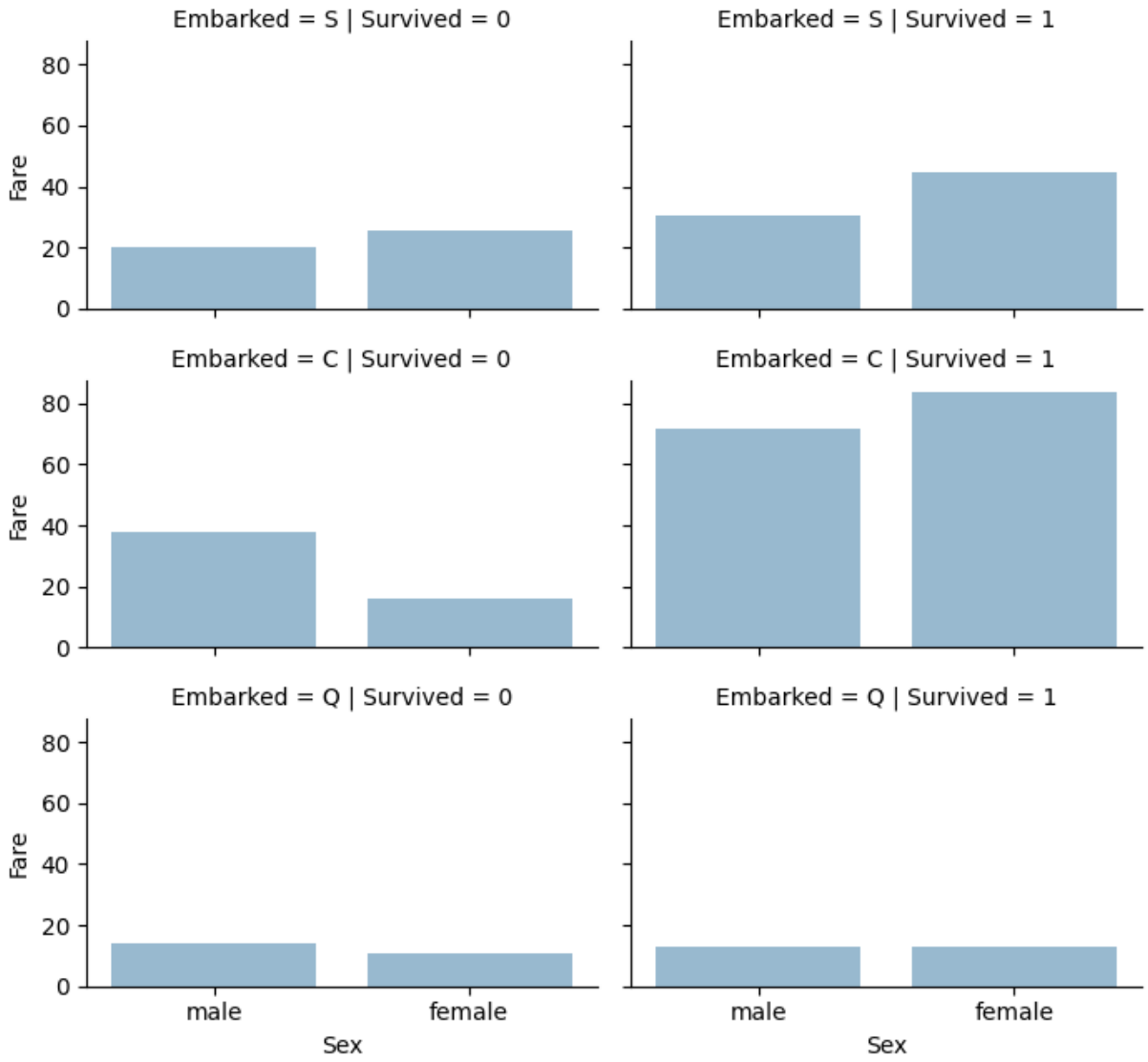

Рисунок 9.18. Графіки співвідношення місця посадки та виживання

```
corr_train = train_data.corr()
sns.heatmap(corr_train)
plt.show()
# Результати свідчать, що стовпці SibSp та Parch видалено, тому ми можемо об'єднати їх для зменшення розмірності набору даних.
```

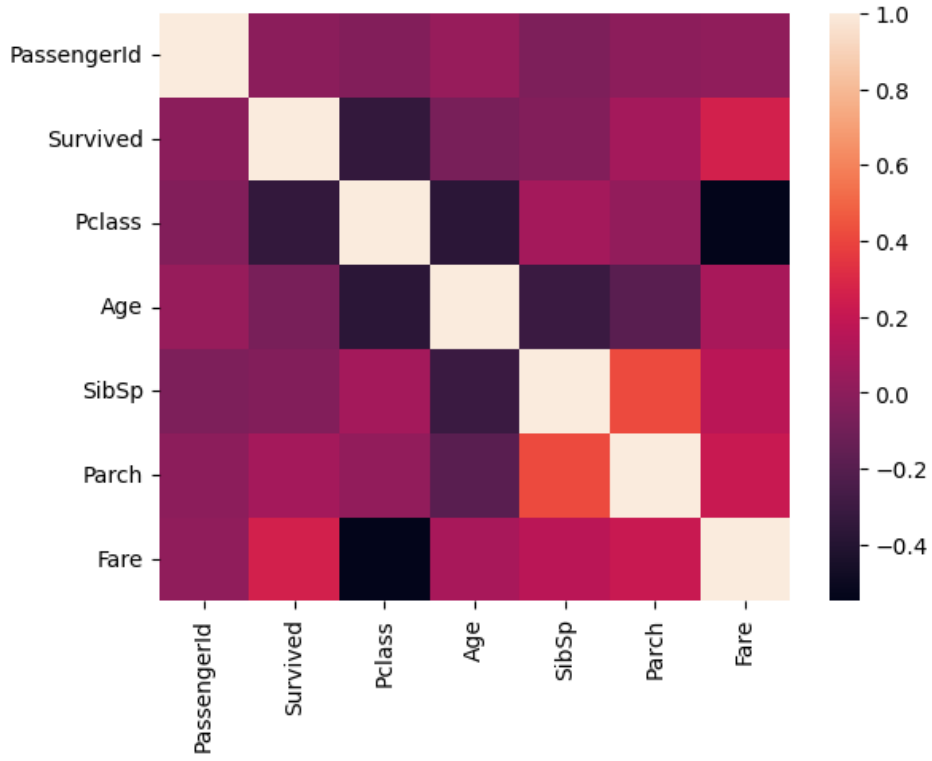

Рисунок 9.19. Теплова карта кореляційного зв’язку

5. Виконаємо простий аналіз даних.

Результати свідчать, що жінки мали близько 74% на виживання, в той час як чоловіки мали 81% на смерть.

```
((train_data.groupby(['Sex','Survived']).Survived.count() * 100) /
train_data.groupby('Sex').Survived.count())
```

```
Sex     Survived
female  0           25.796178
        1           74.203822
male    0           81.109185
        1           18.890815
Name: Survived, dtype: float64
```

Рисунок 9.20. Гендерне розподілення виживання

Результати свідчать, що люди, які належали до третього класу (бідніший), швидше за все, помруть, тоді як люди, які належать до першого класу (багатші), швидше за все, виживуть.

```
(train_data.groupby(['Pclass','Survived']).Survived.count() * 100) /
train_data.groupby('Pclass').Survived.count()
```

```
Pclass  Survived
1       0           37.037037
        1           62.962963
2       0           52.717391
        1           47.282609
3       0           75.763747
        1           24.236253
Name: Survived, dtype: float64
```

Рисунок 9.21. Класове розподілення виживання

Результати свідчать, про те, що люди, які вирушили з порту Southampton, швидше за все, загинуть.

```
(train_data.groupby(['Embarked','Survived']).Survived.count() * 100) /
train_data.groupby('Embarked').Survived.count()
```

```
Embarked  Survived
C         0           44.642857
          1           55.357143
Q         0           61.038961
          1           38.961039
S         0           66.304348
          1           33.695652
Name: Survived, dtype: float64
```

Рисунок 9.22. Географічне розподілення виживання

Результати свідчать, що середній вік людей, які вижили, становив близько 28 років.

```
train_data.groupby(by=['Survived']).mean()["Age"]
```

```
Survived
0    30.626179
1    28.343690
Name: Age, dtype: float64
```

Рисунок 9.23. Середній вік особ, що вижили та померли

6. Виконаємо роботу над NULL значенням.

Перед тим, як заповнити відсутні значення, видалимо стовпець Cabin з обох наборів даних.

```
train_data.drop('Cabin', axis = 1, inplace = True)
test_data.drop('Cabin', axis = 1, inplace = True)

combined_data = [train_data, test_data]
for data in combined_data:
    print(data.isnull().sum())
    print('*' * 20)
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Embarked         2
dtype: int64
********************
PassengerId     0
Pclass          0
Name            0
Sex             0
Age            86
SibSp           0
Parch           0
Ticket          0
Fare            1
Embarked        0
dtype: int64
********************
```

Рисунок 9.24. Визначення кількості NULL значень за стовпцями

Заповнимо стовпчики Age та Fare середніми значеннями, а стовпчик Embarked – значеннями, які зустрічаютсья найчастіше.

```
for data in combined_data:
    data.Age.fillna(data.Age.mean(), inplace = True)
    data.Fare.fillna(data.Fare.mean(), inplace = True)
```

```
# З візуалізацій, нам відомо, що порт Southampton є найпопулярнішим місцем посадки, тому, заповнимо пропущені значення "S"

train_data.Embarked.fillna('S', inplace = True)
```

7. Конвертуємо категоріальні ознаки

Почнемо з перетворення ознаки Sex у категоричні значення female = 1, male = 0

```
def change_gender(x):
    if x == 'male':
        return 0
    elif x == 'female':
        return 1
train_data.Sex = train_data.Sex.apply(change_gender)
test_data.Sex = test_data.Sex.apply(change_gender)

# або використаємо
# train_data.Sex = train_data.Sex.map({'female':1, 'male':0})
```

За допомогою функції map замінимо значення стовпчику Embarked S = 1, C = 2, Q = 0.

```
change = {'S':1,'C':2,'Q':0}
train_data.Embarked = train_data.Embarked.map(change)
test_data.Embarked = test_data.Embarked.map(change)
```

8. Виконаємо вилучення особливостей

Під час візуалізації кореляційної теплової карти ми виявили, що стовпці Sibsp та Parch тісно пов'язані між собою, тому створимо новий стовпець під назвою Alone, використовуючи ці два стовпці -------> 1 = Alone , 0 = not Alone.

```
train_data['Alone'] = train_data.SibSp + train_data.Parch
test_data['Alone'] = test_data.SibSp + test_data.Parch

train_data.Alone = train_data.Alone.apply(lambda x: 1 if x == 0 else 0)
test_data.Alone = test_data.Alone.apply(lambda x: 1 if x == 0 else 0)

# Тепер вилучимо стовпці SibSp та Parch для навчального та тестового наборів даних
train_data.drop(['SibSp','Parch'], axis = 1, inplace = True)
test_data.drop(['SibSp','Parch'], axis = 1, inplace = True )
```

Створимо новий елемент Title назви з існуючого елемента Name.

```
train_data.Name.str.extract(' ([A-Za-z]+)\.', expand=False).unique().size
# всього 17 унікальних назв

# Створимо елемент Title, що буде містити назву пасажира, вилучимо
стовпець Name
for data in combined_data:
    data['Title'] = data.Name.str.extract('([A-Za-z]+)\.', expand = False)
    data.drop('Name', axis = 1, inplace = True)

train_data.Title.value_counts()
```

```
Mr          517
Miss        182
Mrs         125
Master       40
Dr            7
Rev           6
Mlle          2
Major         2
Col           2
Countess      1
Capt          1
Ms            1
Sir           1
Lady          1
Mme           1
Don           1
Jonkheer      1
Name: Title, dtype: int64
```

Рисунок 9.25. Визначення популярності значень Title

```
test_data.Title.unique()
```

```
array(['Mr', 'Mrs', 'Miss', 'Master', 'Ms', 'Col', 'Rev', 'Dr', 'Dona'],
      dtype=object)
```

Рисунок 9.26. Масив значень Title

Замінимо назви, що зустрічаються найменше на Rare.

```
least_occuring = [ 'Don', 'Rev', 'Dr', 'Mme', 'Ms',
       'Major', 'Lady', 'Sir', 'Mlle', 'Col', 'Capt', 'Countess','Dona',
       'Jonkheer']
for data in combined_data:
    data.Title = data.Title.replace(least_occuring, 'Rare')
```

```
# Виконаємо відображення заголовків, замінивши їх на порядкові номери
title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
for data in combined_data:
    data['Title'] = data['Title'].map(title_mapping)
```

Видалимо стовпці PassengerId та Ticket

```
columns_to_drop = ['PassengerId','Ticket']
train_data.drop(columns_to_drop, axis = 1, inplace = True)
test_data.drop(columns_to_drop[1], axis = 1, inplace = True)
```

Об’єднаємо стовпці Age та Fare

```
for dataset in combined_data:
    dataset.loc[ dataset['Age'] <= 16, 'Age'] = 0
    dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1
    dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <= 48), 'Age'] = 2
    dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3
    dataset.loc[ dataset['Age'] > 64, 'Age'] = 4

for data in combined_data:
    data.loc[data['Fare'] < 30, 'Fare'] = 1
    data.loc[(data['Fare'] >= 30) & (data['Fare'] < 50),'Fare'] = 2
    data.loc[(data['Fare'] >= 50) & (data['Fare'] < 100),'Fare'] = 3
    data.loc[(data['Fare'] >= 100),'Fare'] = 4

corr_train = train_data.corr()
sns.heatmap(corr_train)
plt.show()
```

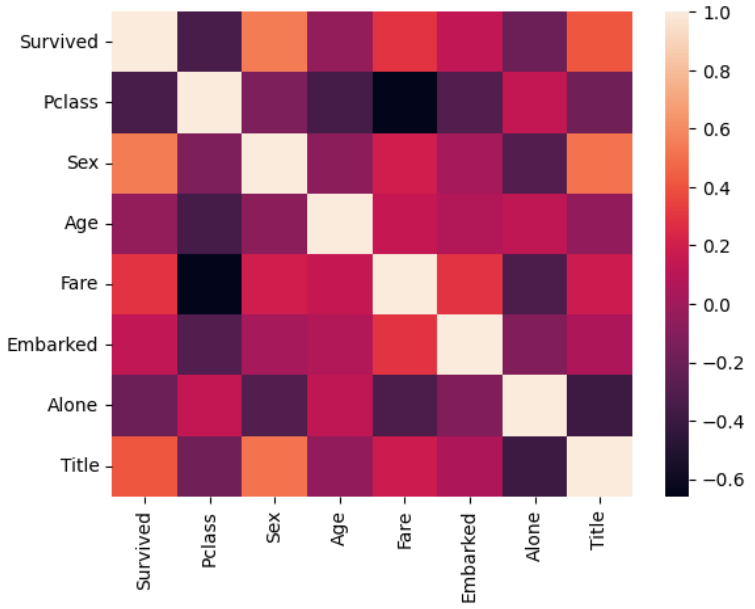

Рисунок 9.27. Результуюча теплова мапа

9. Виконаємо підготовку даних для навчання та тестування

```
X_train = train_data.drop("Survived", axis=1)
Y_train = train_data["Survived"]
X_test = test_data.drop("PassengerId", axis = 1)
print("shape of X_train",X_train.shape)
print("Shape of Y_train",Y_train.shape)
print("Shape of x_test",X_test.shape)
```

10. Розробимо та охарактеризуємо модель нейронної мережі

Примітка. Під час розробки ви можете використовувати різну кількість нейронів для кожного шару, різне значення для відсіву. Ви також можете змінювати значення гіперпараметрів для отримання кращого результату.

```
# Нейронна мережа
model = Sequential([
    Dense(32, input_dim=7, activation='relu'),
    Dense(64, activation='relu', kernel_initializer='he_normal',
use_bias=False),
    tf.keras.layers.BatchNormalization(),
    Dense(128, activation='relu', kernel_initializer='he_normal',
use_bias=False),
    Dropout(0.1),
    Dense(64, activation='relu', kernel_initializer='he_normal',
use_bias=False),
    Dropout(0.1),
    Dense(32, activation='relu'),
    Dropout(0.15),
    Dense(16, activation='relu'),
    Dense(8, activation='relu', kernel_initializer='he_normal',
use_bias=False),
    Dense(1, activation='sigmoid')
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                256

 dense_1 (Dense)             (None, 64)                2048

 batch_normalization (Batch  (None, 64)                256
 Normalization)

 dense_2 (Dense)             (None, 128)               8192

 dropout (Dropout)           (None, 128)               0

 dense_3 (Dense)             (None, 64)                8192

 dropout_1 (Dropout)         (None, 64)                0

 dense_4 (Dense)             (None, 32)                2080

 dropout_2 (Dropout)         (None, 32)                0

 dense_5 (Dense)             (None, 16)                528

 dense_6 (Dense)             (None, 8)                 128

 dense_7 (Dense)             (None, 1)                 9

=================================================================
Total params: 21689 (84.72 KB)
Trainable params: 21561 (84.22 KB)
Non-trainable params: 128 (512.00 Byte)
_________________________________________________________________
```

Рисунок 9.28. Характеристика моделі нейронної мережі

Скомпілюємо та натренуємо модель.

```
model.compile(loss=tf.keras.losses.binary_crossentropy,
              optimizer=tf.keras.optimizers.Adam(),
              metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=32, verbose=2, epochs=50)
```

```
Epoch 43/50
28/28 - 0s - loss: 0.3730 - accuracy: 0.8462 - 151ms/epoch - 5ms/step
Epoch 44/50
28/28 - 0s - loss: 0.3737 - accuracy: 0.8361 - 201ms/epoch - 7ms/step
Epoch 45/50
28/28 - 0s - loss: 0.3668 - accuracy: 0.8339 - 172ms/epoch - 6ms/step
Epoch 46/50
28/28 - 0s - loss: 0.3648 - accuracy: 0.8406 - 195ms/epoch - 7ms/step
Epoch 47/50
28/28 - 0s - loss: 0.3678 - accuracy: 0.8429 - 248ms/epoch - 9ms/step
Epoch 48/50
28/28 - 0s - loss: 0.3677 - accuracy: 0.8361 - 129ms/epoch - 5ms/step
Epoch 49/50
```

Рисунок 9.29. Процес навчання нейронної мережі

11. Прогнозування на основі тестових даних

```
predict = model.predict(X_test)
predict = (predict > 0.5).astype(int).ravel()
print(predict)
```

```
14/14 [==============================] - 1s 4ms/step
[0 1 0 0 1 0 1 0 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 1 1 0 1 0 0 0 0 0 1 1 1 0 0
 0 0 1 0 0 0 1 1 0 1 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1
 1 0 0 1 0 1 1 0 0 0 0 0 1 0 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0
 1 1 1 1 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0
 0 0 1 0 0 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 1
 0 1 0 0 0 0 0 1 0 1 0 1 0 0 1 1 1 1 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0
 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1
 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 1 1 0 0 1 0 0 1 1 0 1 0 0
 1 0 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0
 1 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 0
 0 1 0 0 1 1 0 0 0 0 0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0
 0 1 1 1 1 0 0 1 0 0 1]
```

Рисунок 9.30. Прогнозування «вижив/не вижив»

```
submit = pd.DataFrame({"PassengerId":test_data.PassengerId,
'Survived':predict})
submit.to_csv("final_submission.csv",index = False)

from sklearn import metrics
Y_pred_rand = (model.predict(X_train) > 0.5).astype(int)
print('Precision : ', np.round(metrics.precision_score(Y_train,
Y_pred_rand)*100,2))
print('Accuracy : ', np.round(metrics.accuracy_score(Y_train,
Y_pred_rand)*100,2))
print('Recall : ', np.round(metrics.recall_score(Y_train,
Y_pred_rand)*100,2))
print('F1 score : ', np.round(metrics.f1_score(Y_train,
Y_pred_rand)*100,2))
print('AUC : ', np.round(metrics.roc_auc_score(Y_train,
Y_pred_rand)*100,2))
```

```
28/28 [==============================] - 0s 7ms/step
Precision :  80.42
Accuracy :  84.62
Recall :  79.24
F1 score :  79.82
AUC :  83.61
```

Рисунок 9.31. Результуючі показники моделі

```
# Побудова Confusion Matrix на тепловій карті
matrix = metrics.confusion_matrix(Y_train, Y_pred_rand)
sns.heatmap(matrix, annot = True,fmt = 'g')
plt.show()
```

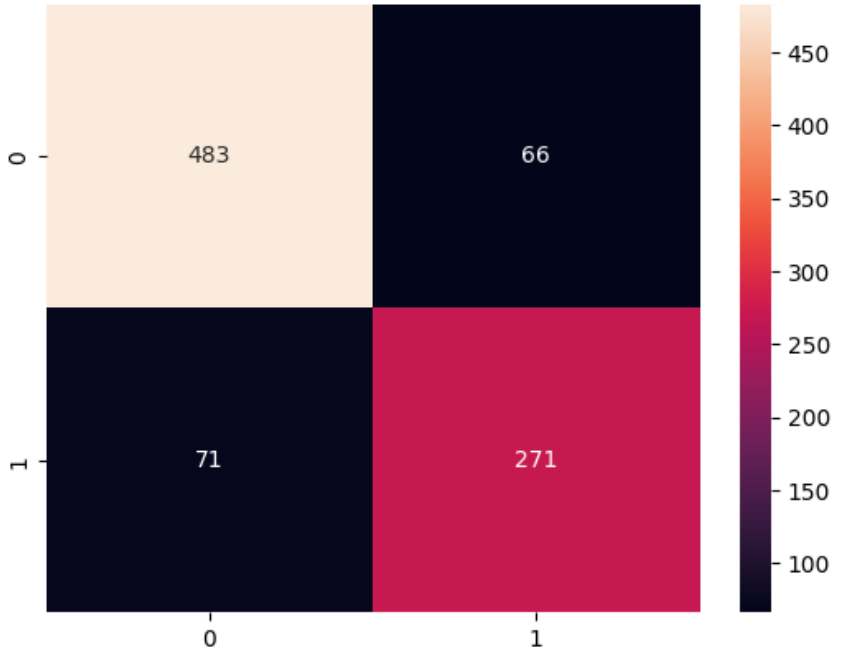

Рисунок 9.32. Confusion Matrix моделі

Контрольне запитання №1

Вкажіть результуюче значення Accuracy.
Відповідь: **84.62**

Контрольне запитання №2

Вкажіть результуюче значення Precision.
Відповідь: **80.42**

Контрольне запитання №3

Вкажіть яке значення Survived приймає PassengerId 900.
Відповідь: …

Висновки: …

Завдання для виконання лабораторної роботи

Титанік – британський пасажирський лайнер компанії White Star Line, який затонув у північній частині Атлантичного океану рано вранці 15 квітня 1912 року після зіткнення з айсбергом під час свого першого рейсу з Саутгемптона до Нью-Йорка з приблизно 2224 пасажирами і членами екіпажу, які перебували на борту, понад 1500 загинули, що робить цю катастрофу однією з найсмертоносніших комерційних морських катастроф мирного часу в сучасній історії. Хоча на борту було близько 2224 пасажирів і членів екіпажу, ми маємо дані про 1300 пасажирів. З цих 1300, близько 900 даних використовуються з навчальною метою, а решта 400 – з тестовою. У цьому завдані вам надано близько 400 тестових даних з відсутнім стовпчиком вижившних. Вам треба змоделювати модель нейронної мережі, щоб передбачити, чи вижили пасажири у тестових даних, чи ні. Як навчальні, так і тестові дані не є чистими (містять багато пропущених значень). Побудуйте модель з найкращою точністю.

Характеристика даних:

- Survival (виживання): 0 – ні, 1 – так;
- Pclass (номер класу): 1-й – вищий, 2-й – середній, 3-й – нижчий;
- sibsp (кількість братів і сестер / подружжя на борту): Sibling – брат, сестра, зведений брат, зведена сестра, Spouse – чоловік, дружина (коханки та наречені не враховуються);
- parch (кількість батьків / дітей на борту): Parent – мати, батько, Child – донька, син, падчерка, пасинок. Деякі діти подорожували лише з нянею, тому для них parch = 0;
- Ticket (квиток): номер квитка;
- Fare (вартість): пасажирський тариф;
- Cabin (каюта): номер каюти;
- Port of Embarkation (порт посадки): C – Cherbourg, Q – Queenstown, S – Southampton;

- Name (ім’я), Sex (стать), Age (вік).

Примітка. Дана лабораторна робота не має конкретних варіантів, але передбачає, що виконання буде базуватися на принципах креативності та зацікавленості здобувачів. Додаткову інформацію та приклади реалізації можна знайти на сайті Kaggle.

Посилання на навчальний та тестовий набори даних

Контрольне запитання №1

Вкажіть результуюче значення Accuracy.
Відповідь: _____

Контрольне запитання №2

Вкажіть результуюче значення Precision.
Відповідь: _____

Контрольне запитання №3

Вкажіть яке значення Survived приймає PassengerId 900.
Відповідь: _____

Welcome to Google Colab

https://colab.research.google.com/drive/1hcaHTcz4n5gMoFiTrfEjHF2sMuBCpN1f?usp=sharing

Головне про Kaggle: **Kaggle** – це найбільша у світі платформа для змагань з даними, де спільнота з понад мільйоном користувачів, включаючи дослідників, інженерів з даними, та статистиків, співпрацюють для вирішення складних наукових завдань. Заснована у 2010 році, Kaggle надає компаніям, організаціям та дослідникам можливість опублікувати свої дані та поставити перед спільнотою виклики у вигляді змагань з призовими фондами. Учасники можуть взяти участь у змаганнях для побудови моделей прогнозування, аналізу даних та розробки алгоритмів, використовуючи машинне навчання та глибоке навчання.

За додатковою інформацією щодо Kaggle та задачі «Titanic» звертайтеся до https://www.kaggle.com/competitions/titanic