Final Project presentation

ML25-Team05

112703003 資訊二 黃柏淵
112703028 資訊二 吳亭翰
112703037 資訊二 陳凱廷
112703046 資訊二 卓俊瑋

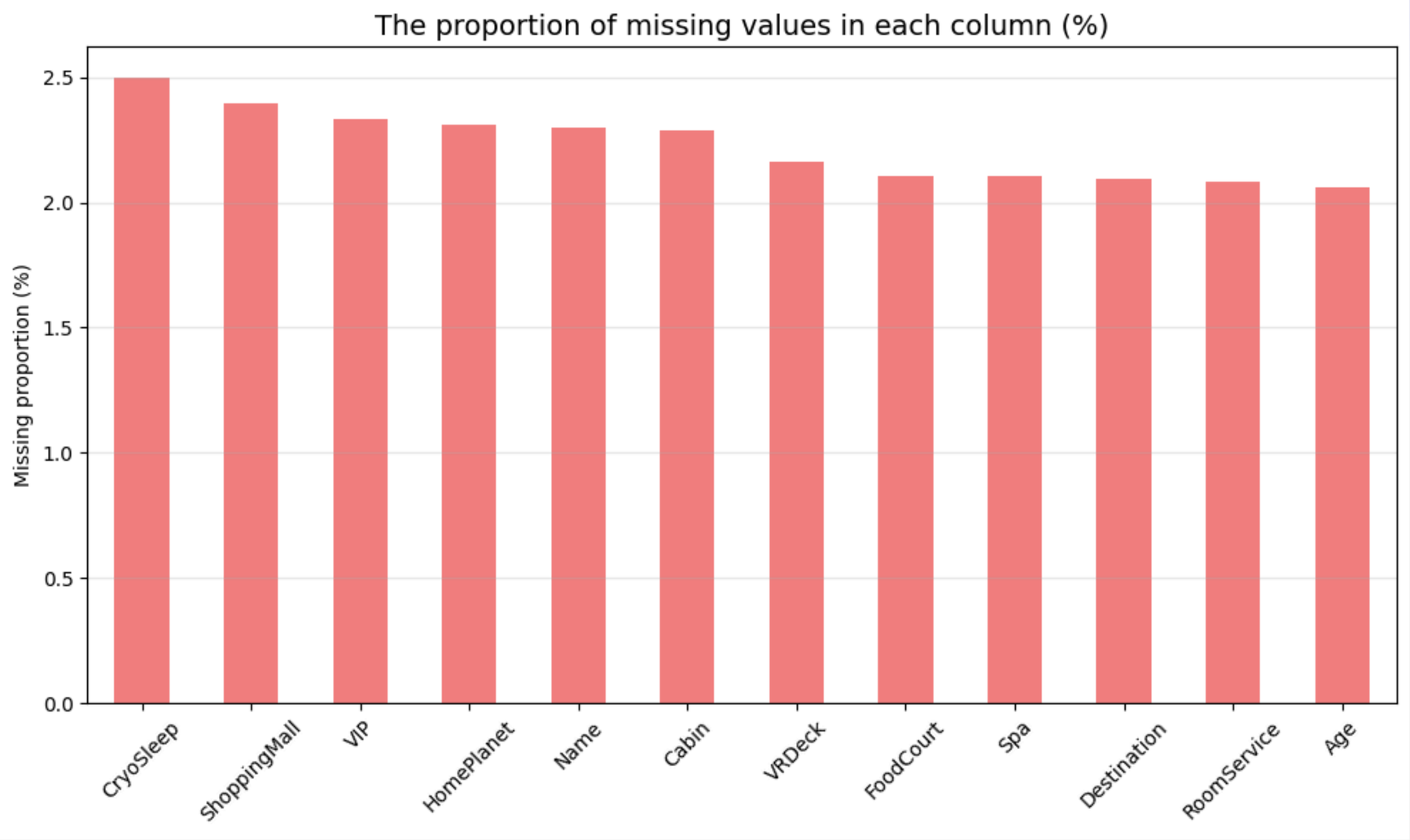
Data

特徵名稱	繁體中文解釋
PassengerId	乘客編號。每位乘客的唯一識別碼，格式為 `gggg_pp`，其中 `gggg` 表示乘客所屬的團體編號，`pp` 表示該團體內的個人編號。乘客可能與家人或朋友一起旅行，也可能獨自旅行。
HomePlanet	家鄉星球。乘客離開的星球，通常是他們的永久居住地，可能影響其旅行目的或行為。常見值包括 Earth（地球）、Europa（歐羅巴）、Mars（火星）等。
CryoSleep	冷凍睡眠。指示乘客是否選擇在航行期間進入冷凍睡眠狀態（True/False）。冷凍睡眠的乘客會被限制在艙內，可能影響其消費行為或生存機率。
Cabin	艙位號碼。乘客所在的艙位，格式為 `deck/num/side`，其中 `deck` 是甲板（字母，如 A、B、C 等），`num` 是艙位號碼，`side` 是側邊（P 為 Port 左舷，S 為 Starboard 右舷）。艙位可能與安全性或舒適度相關。
Destination	目的地。乘客計劃前往的星球，可能是三個可居住的系外行星之一，如 TRAPPIST-1e、55 Cancri E 或 PSO J318.5-22，可能影響旅行動機或準備。
Age	年齡。乘客的年齡（以歲為單位），可能影響體力、決策或被傳送的機率（例如，兒童或老人可能有不同風險）。

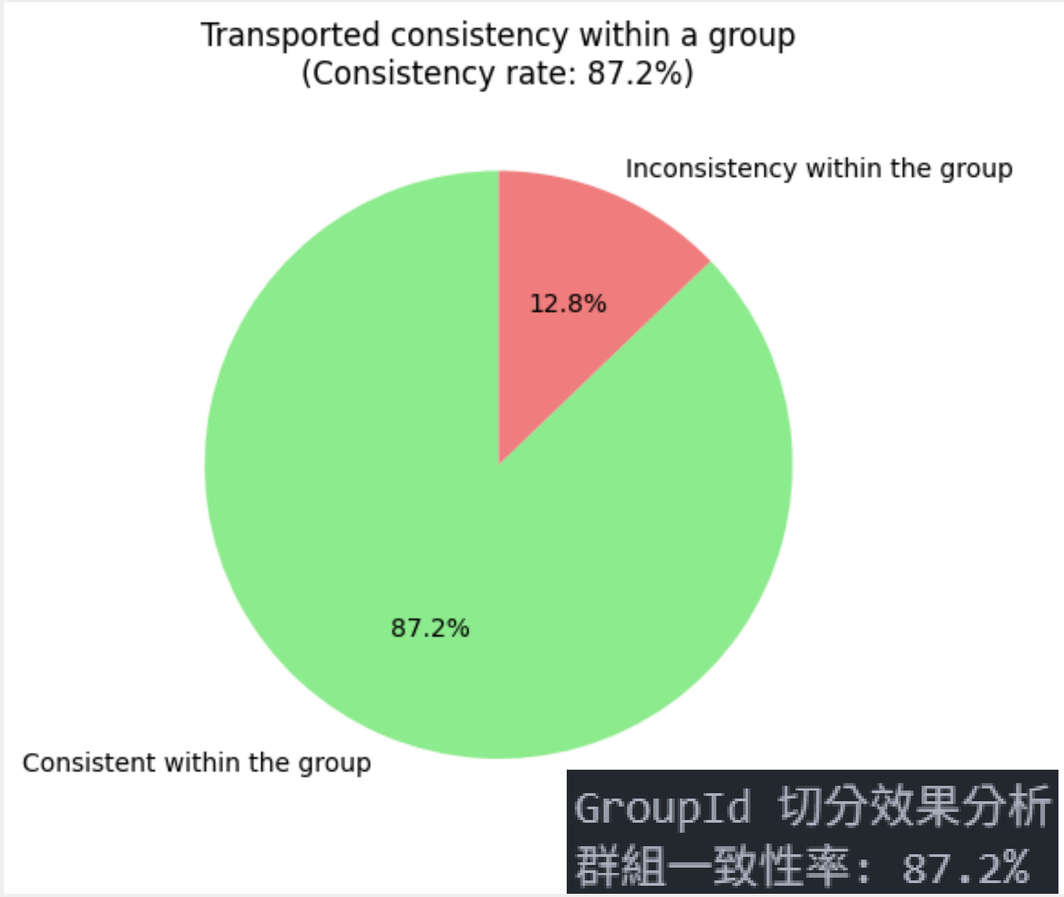
Data

VIP	VIP 服務。指示乘客是否支付了額外的 VIP 服務（True/False），可能與特殊待遇、艙位位置或資源獲取有關。
RoomService	客房服務費用。乘客在客房服務上的消費金額（以貨幣單位計），反映其在航行中的活動水平或奢華程度。
FoodCourt	美食廣場費用。乘客在飛船美食廣場的消費金額，可能顯示其飲食習慣或經濟能力。
ShoppingMall	購物中心費用。乘客在飛船購物中心的消費金額，反映其購物行為或額外需求。
Spa	溫泉費用。乘客在飛船溫泉或放鬆設施上的消費金額，可能與其放鬆需求或生活方式相關。
VRDeck	虛擬實境甲板費用。乘客在虛擬實境娛樂設施上的消費金額，顯示其娛樂偏好或活躍程度。
Name	姓名。乘客的姓名（包括名字和姓氏），可能用於推測家庭關係或社會背景，但通常含缺失值。
Transported	是否被傳送（目標變量）。指示乘客是否被時空異常傳送到另一個維度（True/False）。這是訓練數據集中的目標欄位，用於模型預測，測試數據集中不存在。

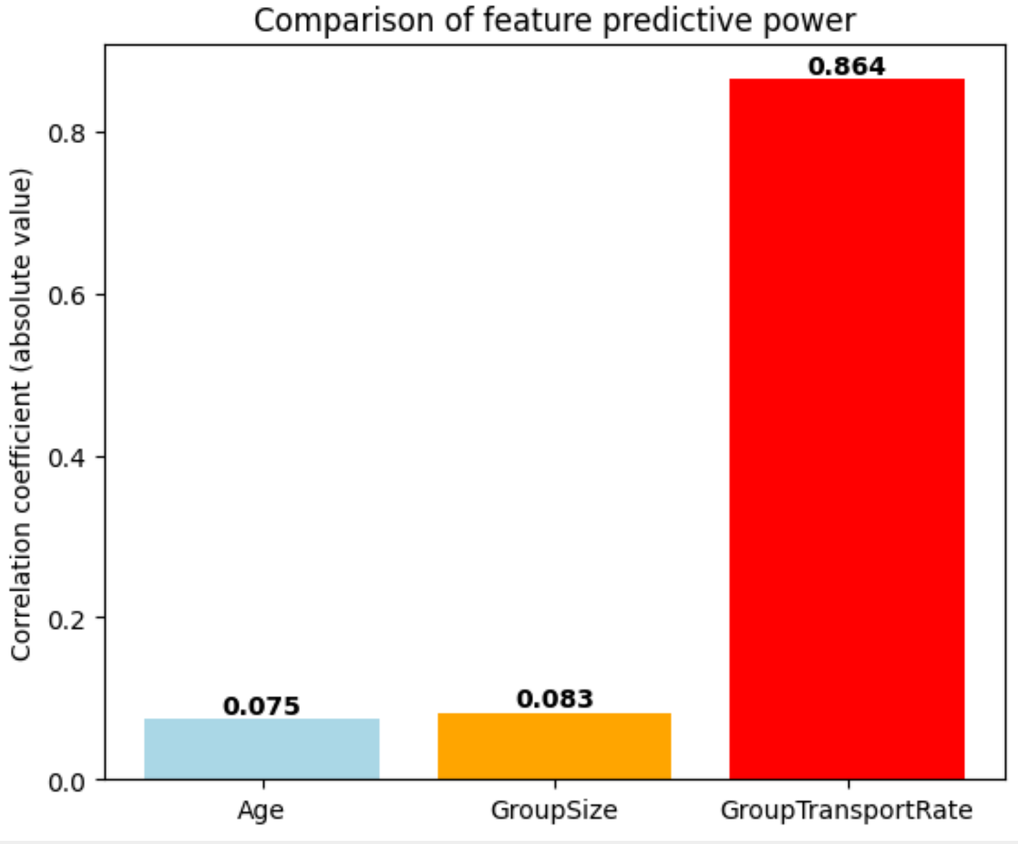
找尋缺失值


The proportion of missing values in each column (%)
Missing proportion (%)
2.5
2.0
1.5
1.0
0.5
0.0
CryoSleep
ShoppingMall
VIP
HomePlanet
Name
Cabin
VRDeck
FoodCourt
Spa
Destination
RoomService
Age

探索資料特性




```
GroupId 切分效果分析
群組一致性率: 87.2%
```



探索資料特性

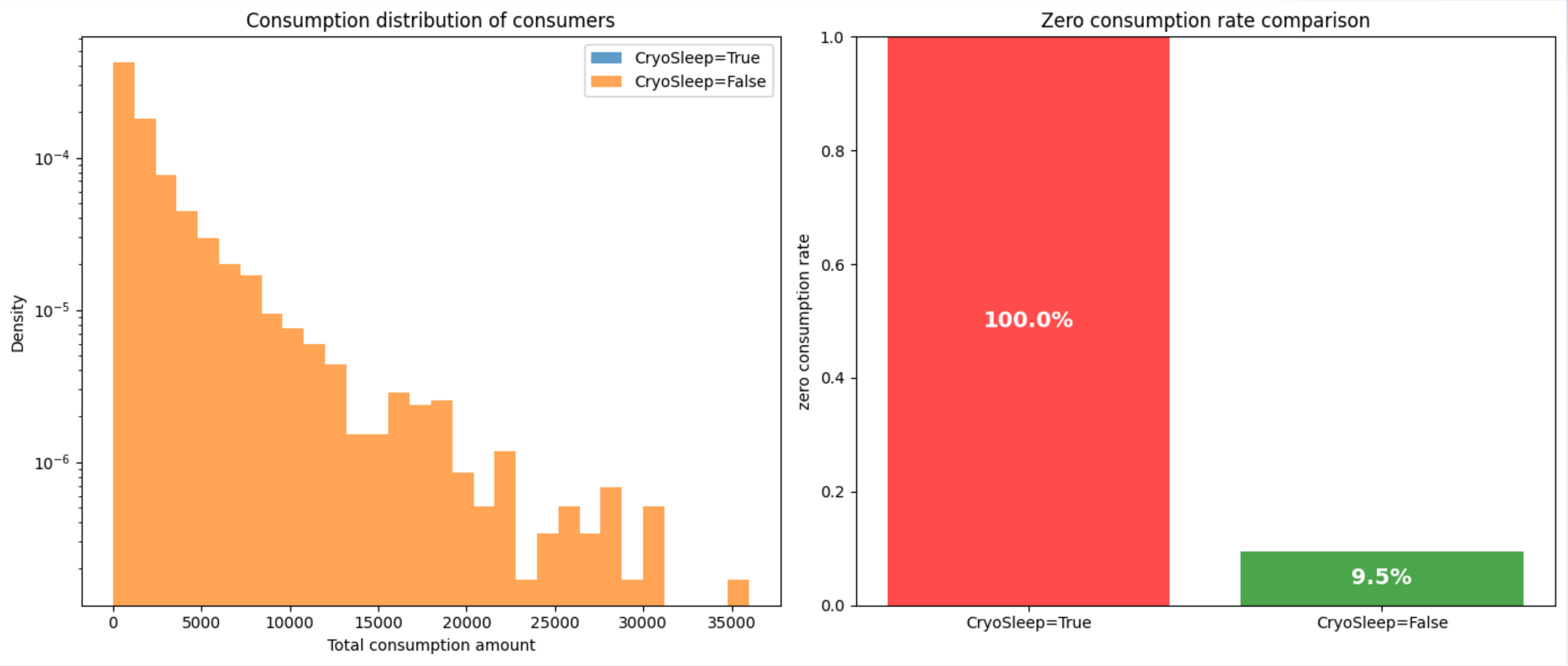
```
=== GroupId 與 HomePlanet 關聯性分析 ===
HomePlanet 在群組內的一致性:
完全一致的群組: 6107 / 6217
一致率: 0.982

=== GroupId 與 Cabin 關聯性分析 ===
Deck 在群組內的一致性:
完全一致的群組: 5697 / 6217
一致率: 0.916

=== GroupId 與 Destination 關聯性分析 ===
Destination 在群組內的一致性:
完全一致的群組: 5397 / 6217
一致率: 0.868
```

填補缺失值

```
# 提取 GroupId 來自 PassengerId
combined['GroupId'] = combined['PassengerId'].str.split("_").str[0]

# 根據 GroupId 填補 HomePlanet、Cabin 和 Destination 的缺失值
combined['HomePlanet'] = combined.groupby('GroupId')['HomePlanet'].transform(
    lambda x: x.fillna(x.mode()[0] if not x.mode().empty else 'Unknown'))
combined['Destination'] = combined.groupby('GroupId')['Destination'].transform(
    lambda x: x.fillna(x.mode()[0] if not x.mode().empty else 'Unknown'))
combined['Cabin'] = combined.groupby('GroupId')['Cabin'].transform(
    lambda x: x.fillna(x.mode()[0] if not x.mode().empty else 'Unknown'))
```

- 當乘客在同一個group時，他們有98%是來自同一個星球，同時他們待在同一個cabin的比例大約是91.6%，目的地相同的比例為88.27%

探索資料特性


Consumption distribution of consumers
CryoSleep=True
CryoSleep=False
Density
10^{-4}
10^{-5}
10^{-6}
0
5000
10000
15000
20000
25000
30000
35000
Total consumption amount
Zero consumption rate comparison
zero consumption rate
1.0
0.8
0.6
0.4
0.2
0.0
100.0%
9.5%
CryoSleep=True
CryoSleep=False

特徵工程

```
# 根據 CryoSleep 填補消費類別缺失值
spending_columns = ['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']
for col in spending_columns:
    combined[col] = combined.apply(
        lambda row: 0 if pd.isna(row[col]) and row['CryoSleep'] == 'True' else
        row[col] if not pd.isna(row[col]) else
        combined[col].median(), axis=1)
```

`CryoSleep` - Indicates whether the passenger elected to be put into suspended animation for the duration of the voyage. Passengers in cryosleep are confined to their cabins.

探索資料特性

```
=== 個別特徵預測力比較 ===
RoomService vs Transported: -0.2446
FoodCourt vs Transported: 0.0466
ShoppingMall vs Transported: 0.0101
Spa vs Transported: -0.2211
VRDeck vs Transported: -0.2071
TotalSpend vs Transported: -0.1995

總比較:
TotalSpend 相關性: 0.1995
TotalSpend 提升: -18.4%
```

```
=== VIP 欄位基本分析 ===
VIP 原始資料類型: object
VIP 值分布:
VIP
False    8291
NaN       203
True      199
Name: count, dtype: int64
VIP 缺失率: 0.023
```

特徵工程

```
# --- 提取 Cabin 資訊 ---
combined[["Deck", "CabinNum", "Side"]] = combined["Cabin"].str.split("/", expand=True)
combined["CabinNum"] = pd.to_numeric(combined["CabinNum"], errors='coerce')  # 轉為數值

# --- 處理 VIP 特徵 ---
combined['VIP'] = combined['VIP'].fillna('False').astype(str)
```

- 將Cabin拆開成Deck / CabinNum / Side，便於模型使用
- VIP=True的比例僅2.28%，故填補VIP為False

特徵工程

```
# --- 處理其他類別型欄位 ---
combined['CryoSleep'] = combined['CryoSleep'].astype(str)
combined['Deck'] = combined['Deck'].fillna('Unknown').astype(str)
combined['Side'] = combined['Side'].fillna('Unknown').astype(str)
```

- 將其他類型的缺失值補為Unknown，便於之後的補齊

特徵工程

```
# --- 數值型與類別型特徵欄位 ---
numerical_cols = ["Age", "RoomService", "FoodCourt", "ShoppingMall", "Spa", "VRDeck", "CabinNum"]
categorical_cols = ["HomePlanet", "Destination", "Deck", "Side", "CryoSleep", "VIP"]
```

- 數值型： Age、五項花費、CabinNum
- 分類型：HomePlanet、Destination、Deck、Side、CryoSleep、VIP

填補特徵值與編碼

```
# 數值型特徵的處理：填補為中位數，然後進行標準化
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

# 類別型特徵的處理：填補為 'Unknown'，然後進行 OneHotEncoder
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='Unknown')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
```

- 數值型的資料用同一列的中位數補齊，再做標準化
- 分類型的資料填入Missing，再進行編碼

特徵處理

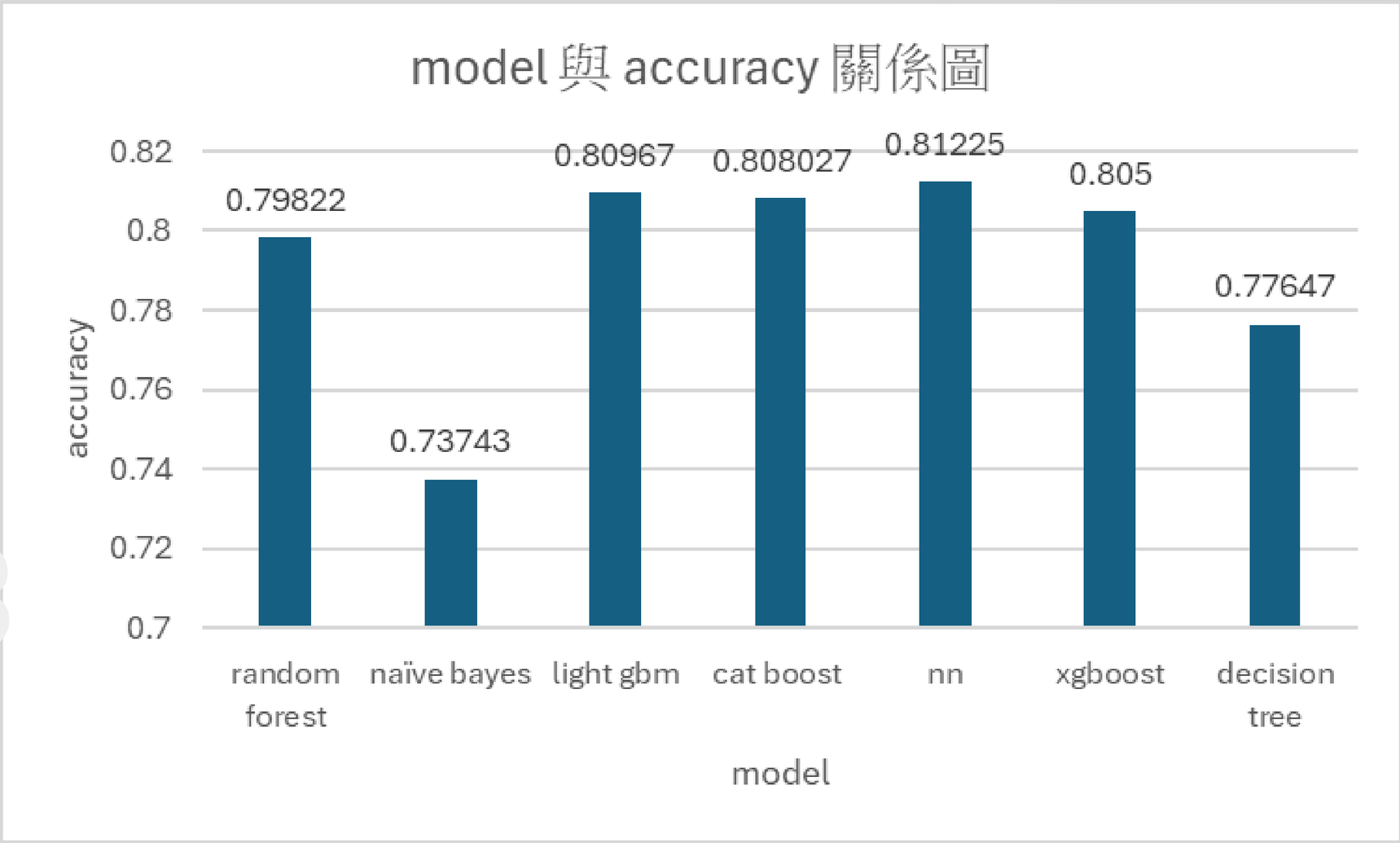
```
# 使用 ColumnTransformer 將不同處理方式應用於不同欄位
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ],
    remainder='drop'  # 忽略未指定的欄位
)
```

- 建立preprocessor，對不同的類別進行轉換

特徵處理

```
# 分割回訓練與測試集
train_processed = combined[combined["is_train"] == 1]
test_processed = combined[combined["is_train"] == 0]

# 特徵欄位與目標欄位
X = train_processed[numerical_cols + categorical_cols]
y = train_processed["Transported"].astype(int)
X_test = test_processed[numerical_cols + categorical_cols]

# --- 整合特徵處理與 Keras 模型訓練 ---
# 在訓練模型前，先處理特徵
X_transformed = preprocessor.fit_transform(X)
X_test_transformed = preprocessor.transform(X_test)
```

Algorithm


model 與 accuracy 關係圖
0.82
0.8
0.78
0.76
0.74
0.72
0.7
accuracy
0.79822
0.73743
0.80967
0.808027
0.81225
0.805
0.77647
random forest
naïve bayes
light gbm
cat boost
nn
xgboost
decision tree
model

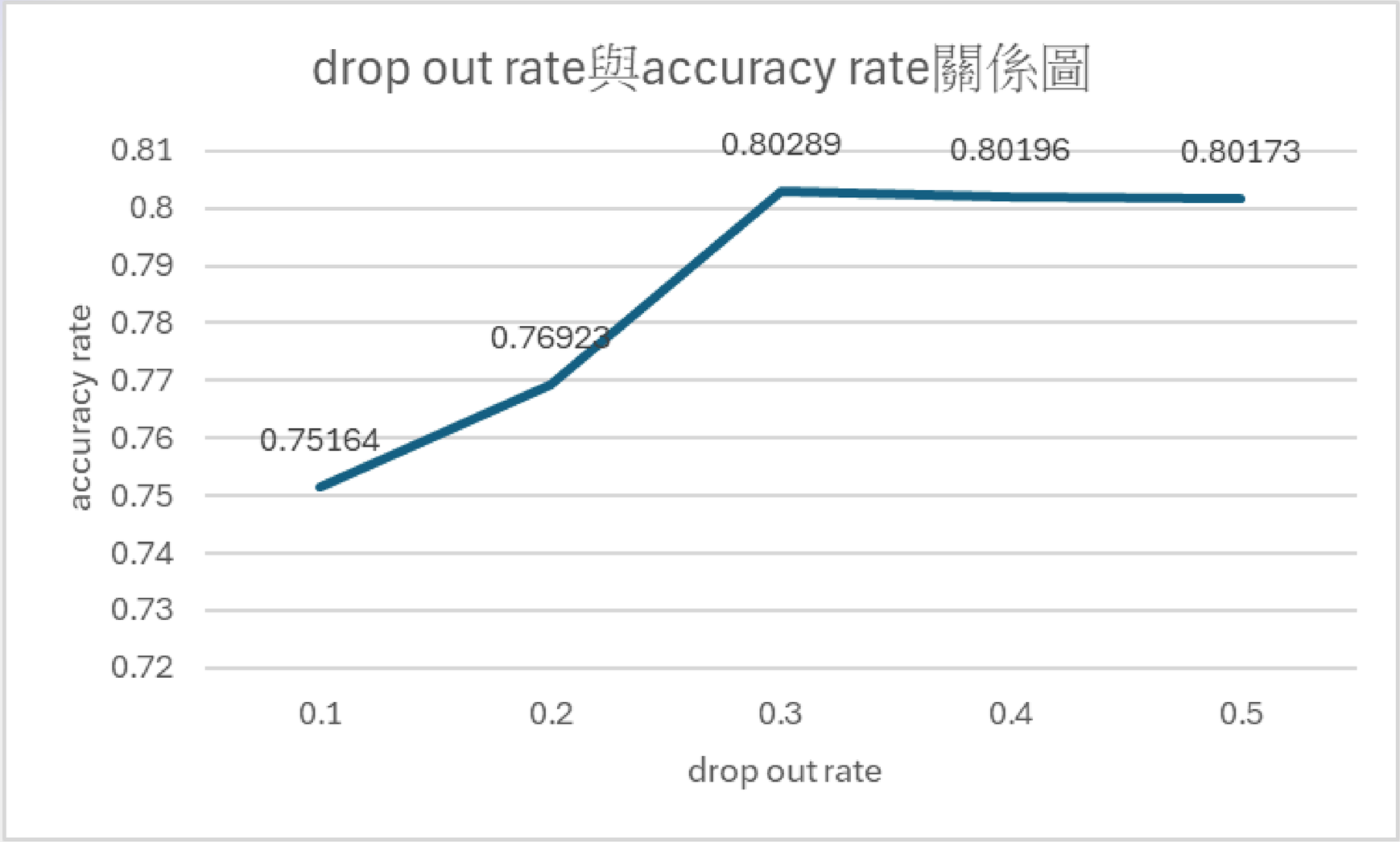
Algorithm

```
# 建立神經網路模型
model = Sequential()

# 輸入層與第一隱藏層，使用 64 個神經元，並添加 Dropout 層以防止過擬合
model.add(Dense(units=64, input_dim=X_transformed.shape[1], activation='relu'))
model.add(Dropout(0.3))

# 第二隱藏層
model.add(Dense(units=32, activation='relu'))
model.add(Dropout(0.3))

# 輸出層：二分類問題，使用 sigmoid 激活函數
model.add(Dense(units=1, activation='sigmoid'))

# 編譯模型，使用 Adam 優化器與二元交叉熵損失函數
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])
```

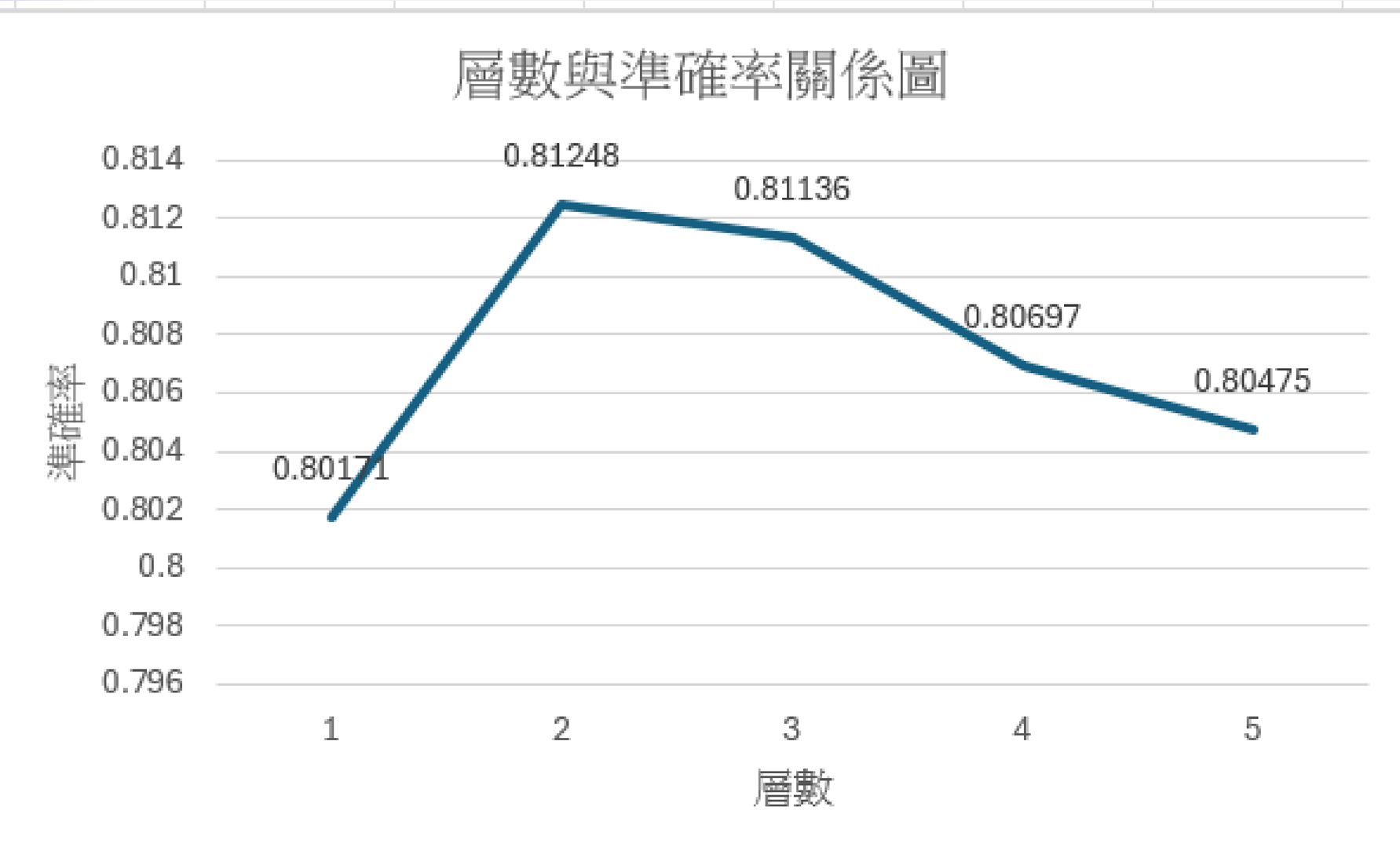
Algorithm

1. sequential neural network model
2. **two** hidden layer
3. use **ReLu** as activation function
4. use Dropout to avoid overfitting
5. use **Adam** optimizer to compile

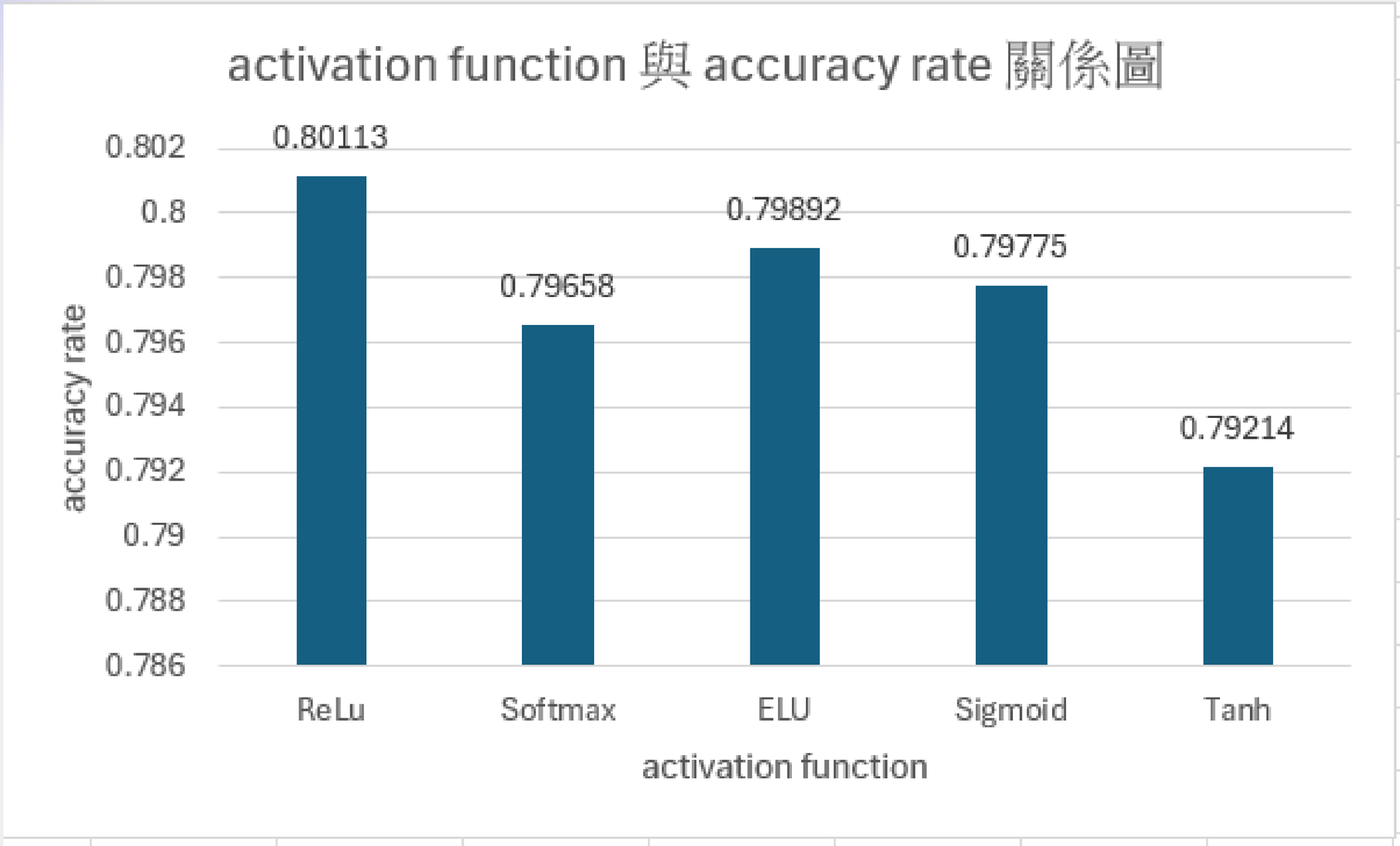
why set drop rate = 0.3 ?



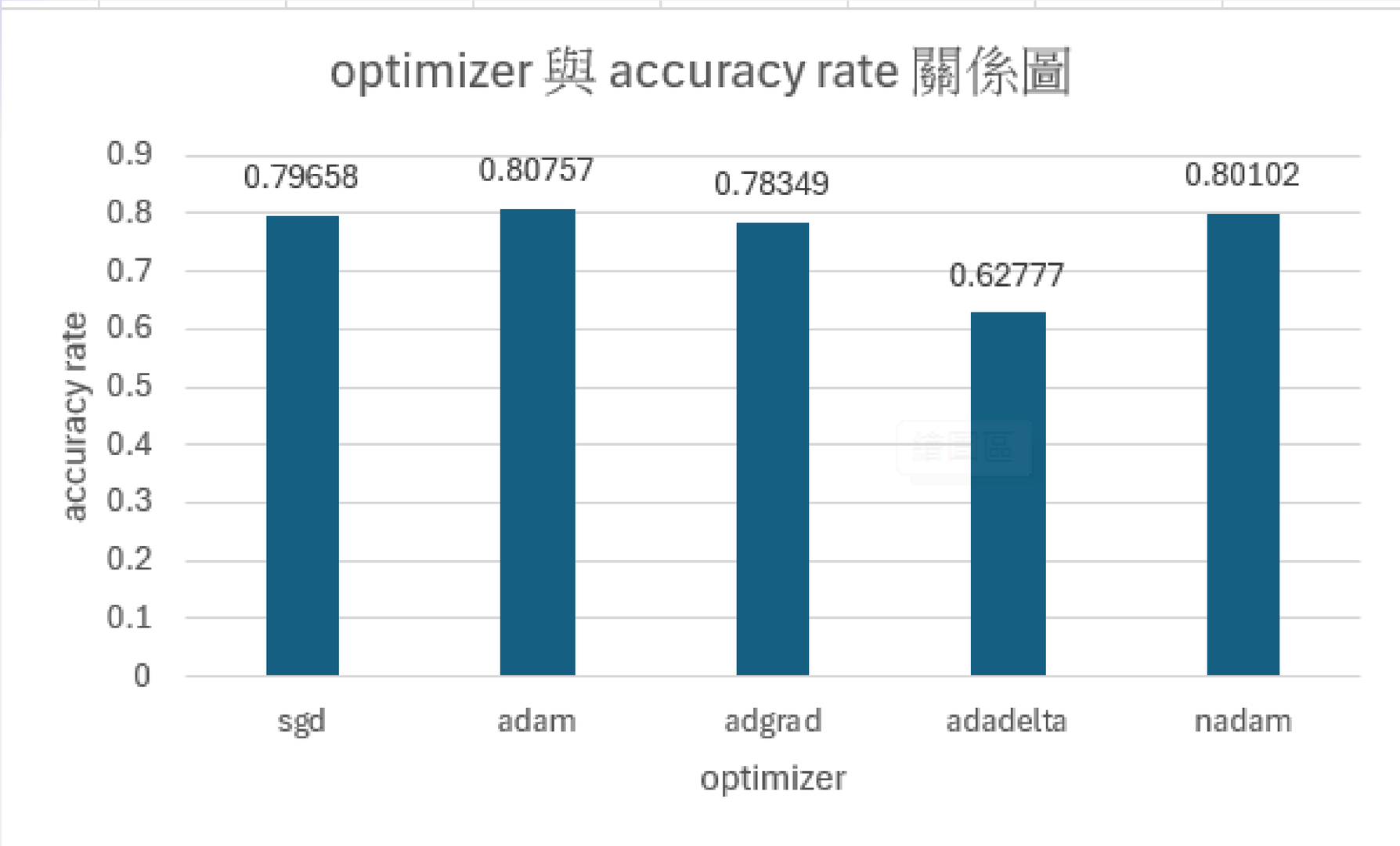
why 2 hidden layers?



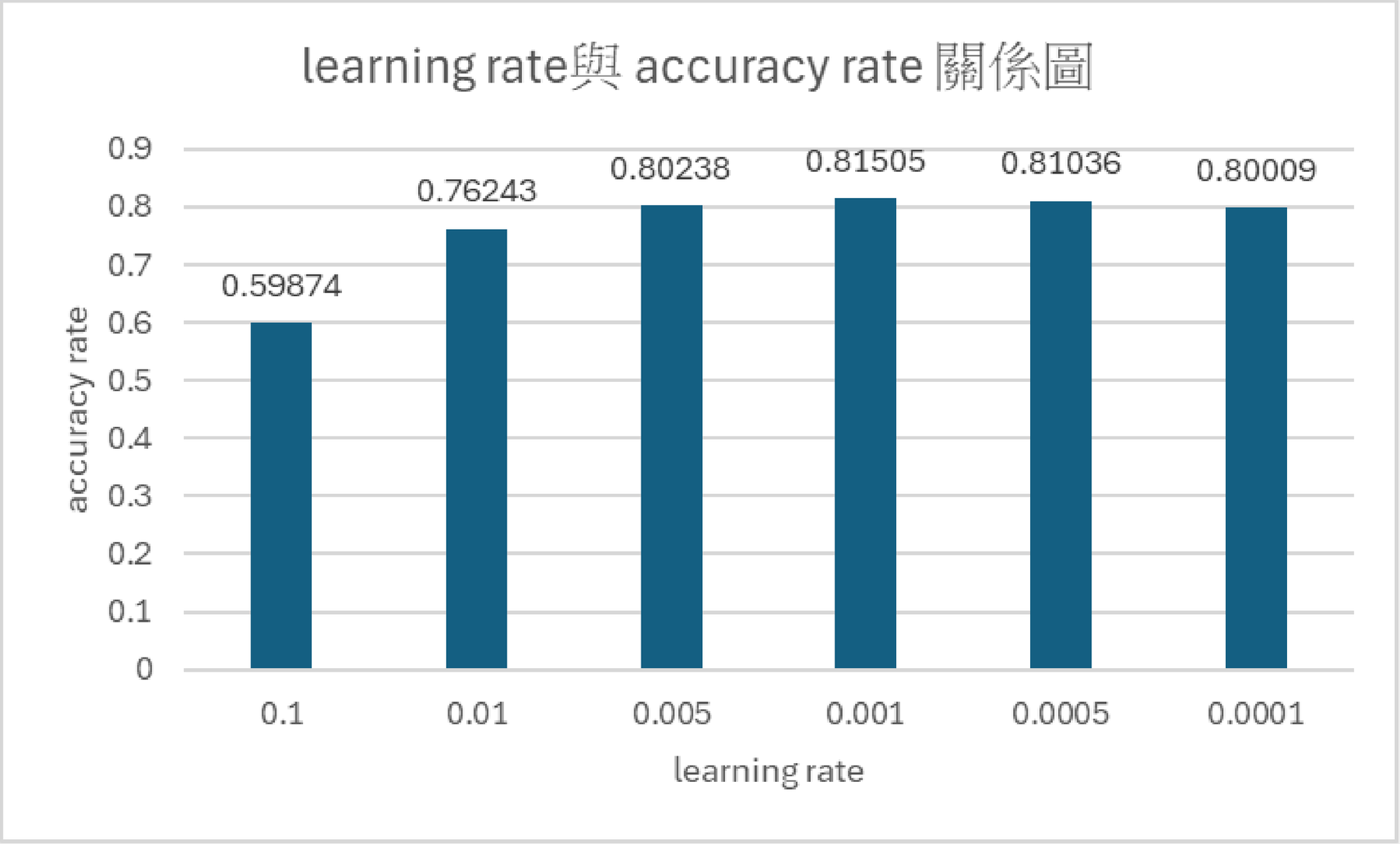
why choosing ReLu?



why choosing Adam?



why set learning rate = 0.001?



訓練模型與預測

```python
# 訓練模型
history = model.fit(X_transformed, y, epochs=50, batch_size=32, validation_split=0.2, verbose=1)

# 預測測試集
predictions = model.predict(X_test_transformed)
predictions = (predictions > 0.5).astype(int)  # 將機率轉為 0 或 1

# 輸出結果
submission = pd.DataFrame({
    "PassengerId": passenger_ids,
    "Transported": predictions.flatten().astype(bool)  # 確保結果為布林值
})
```

競賽結果

32	ML25-Team05		0.81505	71	2m

Thank You