巨量資料分析期末報告

01057051 陳俞君

一. 目的:這個主題想要達成什麼成果

主要目的是利用機器學習技術來預測個體的收入水平是否超過 50K 美元。通過分析人口統計數據和其他特徵，建構分類模型，來識別哪些個體的收入可能會超過此門檻。

二. 文獻回顧:

這個問題以前的人怎麼做，目前有沒有處理的方法

在研究分析中一般利用統計方法和機器學習技術來進行收入預測。比較早期的分析多基於迴歸分析，通過分析人口統計學特徵來預測收入水平。隨著機器學習技術的發展，越來越多研究開始使用新的分類算法，如決策樹、隨機森林、支持向量機、K 近鄰算法和梯度提升等方法，來提高預測的準確性和效率。

本次由機器學習平台 Kaggle 入門(主要是鐵達尼號生存預測)，使用多種 Python lib 撰寫程式來進行數據處理、機器學習建模和數據可視化。。

三. 分析過程及方法:

資料從哪裡取得，我拿裡面那些資料來做，怎麼做，有沒有遇到困難，怎麼克服

資料來源

Income Predictor Dataset- US Adult

使用的數據集來自於 UCI 機器學習資料庫中的 Adult 數據集。該數據集包含了許多個體的人口統計信息和收入水平標籤。具體特徵包括**年齡、工作類別、教育程度、婚姻狀況、職業、種族、性別、每週工作時數等。**

預處理過程

數據清洗：首先，對數據中的缺失值進行處理，將包含缺失值的行刪除並對一些文本數據進行清理，去除多餘的空格。

特徵工程：使用 pd.get_dummies 方法將分類變量轉換為多個二元變量（one-hot encoding），以便機器學習算法可以處理這些數據。

特徵縮放：使用 StandardScaler 對數值特徵進行標準化，以確保各特徵的值在同一範圍內，有助於模型的收斂。

統計分析

對於每個特徵進行了基本的統計描述，包括平均值、中位數、標準差、最小值、最大值等。這有助於了解數據的基本特性，例如：年齡分佈、每週工作小時數的分佈、各類工作類別和教育程度的分佈。

此外，繪製各個特徵的分佈圖，如直方圖和盒鬚圖，以便更直觀了解數據的分佈情況。

模型訓練與評估

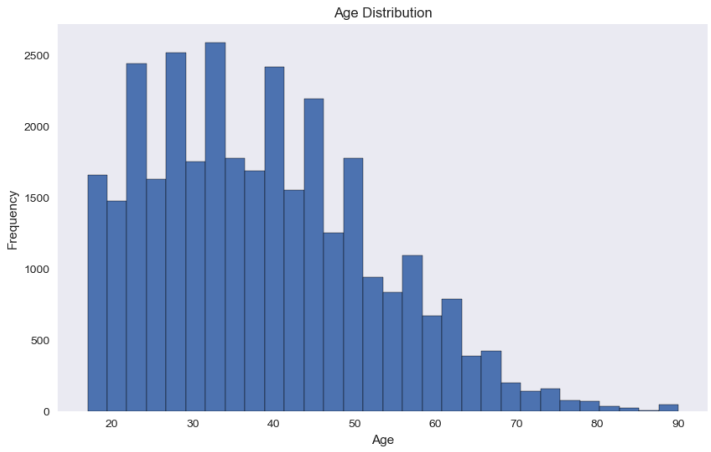
1. 使用多種機器學習算法來訓練模型，包括隨機森林、決策樹、支持向量機、K 近鄰算法和

XGBoost。

2. 使用準確率、混淆矩陣和 ROC 曲線等指標評估這些模型的性能。

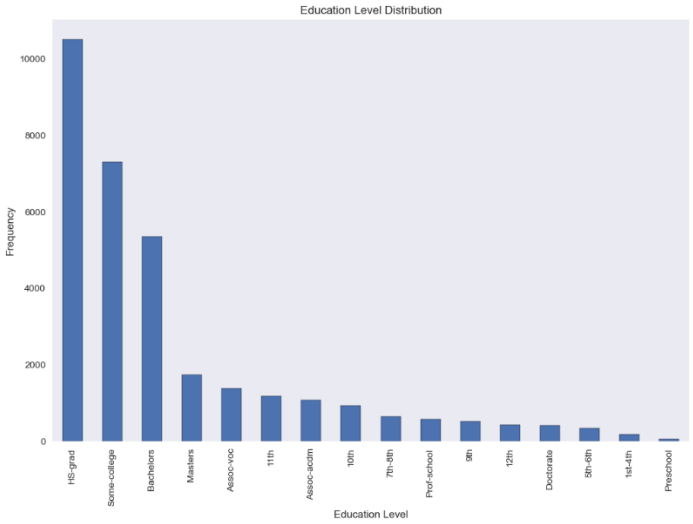
描述性統計分析：

繪製數據分佈圖，如直方圖和盒鬚圖。



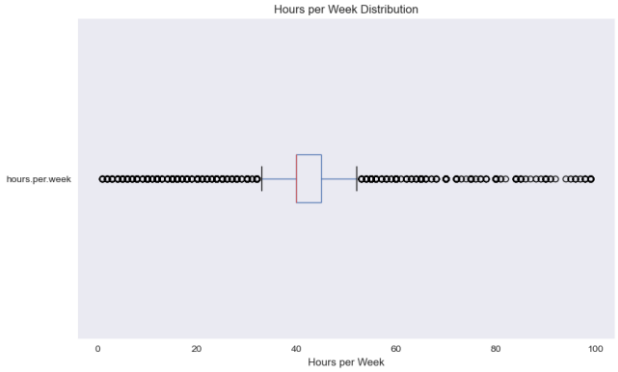
年齡分佈直方圖：展示了不同年齡段的數據分佈情況。

主要分布在 20~45 歲



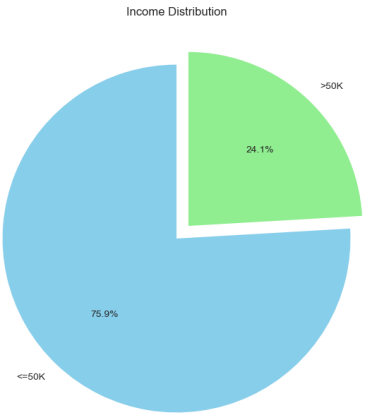
教育程度分佈條形圖：展示了不同教育程度的人數分佈。

最高的前三類別由高到低為高中畢業、大學(學院)、學士



每周工作小時數的盒鬚圖：展示了每周工作小時數的分佈和異常值。

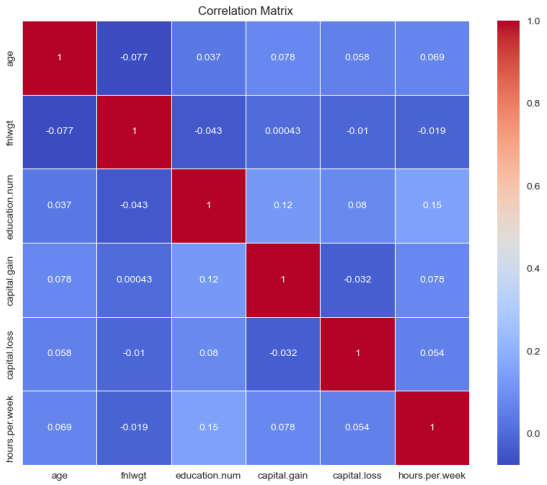
平均在每周 42 小時左右



收入分佈圓餅圖：展示了收入在兩個範圍（<=50K 和 >50K）中的比例。

<=50K 的人佔 75% >50K 的人佔約 25%

相關性分析：計算不同變量之間的相關性。



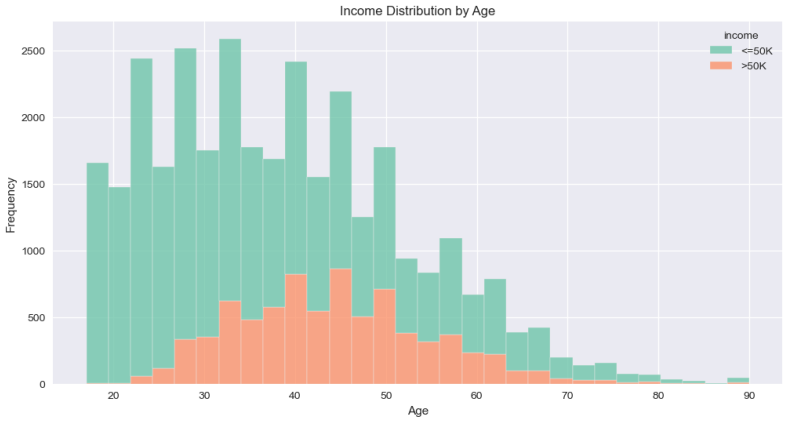
熱力圖展示了數據集中主要數值變量之間的相關性。從圖中可以看出：

變量之間沒有明顯的線性關係。

分群分析 根據收入（<=50K 或 >50K）進行分群，分析各群體的特徵

1. 收入與年齡的關係: 將年齡分段，並分析每個年齡段的收入分佈情況。

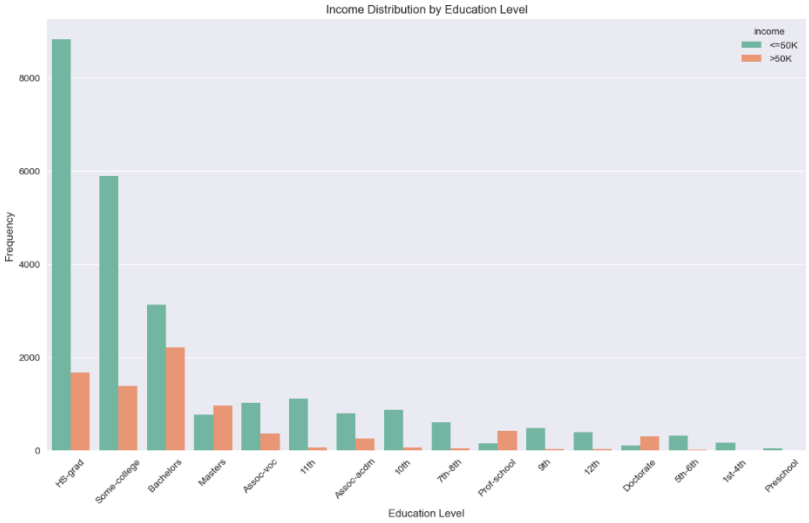
年齡較大的群體中，高收入（>50K）的人數比例相對較高。



2. 收入與教育程度的關係 分析不同教育程度的收入分佈情況。

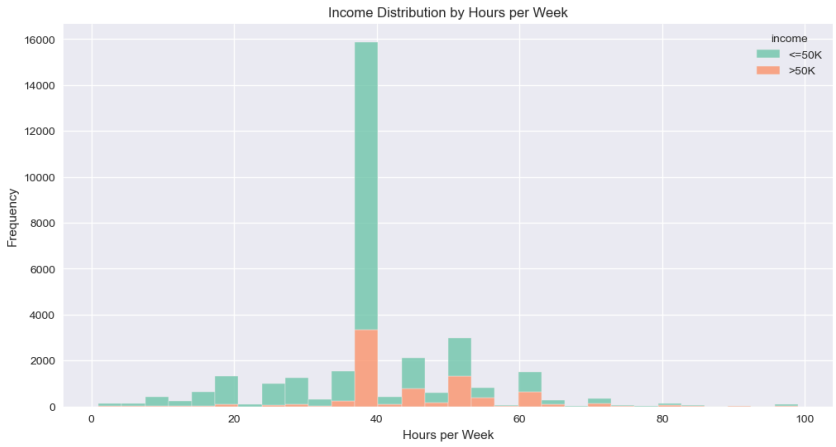
i. 擁有較高教育程度（如學士、碩士及以上學位）的人群中，高收入者比例較高。

ii. 教育程度較低（如未完成高中教育）的群體中，低收入（<=50K）的人數比例較高。



3. 收入與每周工作小時數的關係 每周工作小時數與收入的分佈圖。

每周工作小時數較多的人群中，高收入者比例較高。



四. 結果:

我得到什麼樣的結果，跟我的預期有沒有差距

遇到的困難與克服方法

數據缺失：數據集中存在部分缺失值，刪除包含缺失值的行來清理數據。

特徵處理：分類變量需要進行 one-hot encoding，這會導致特徵數量增加。使用了 pd.get_dummies 方法來自動完成這一過程。

模型過擬合：一些模型（如決策樹）容易過擬合。透過調整模型參數（如最大深度）和使用集成方法（如隨機森林和 XGBoost）來減少過擬合的影響。

相關性分析

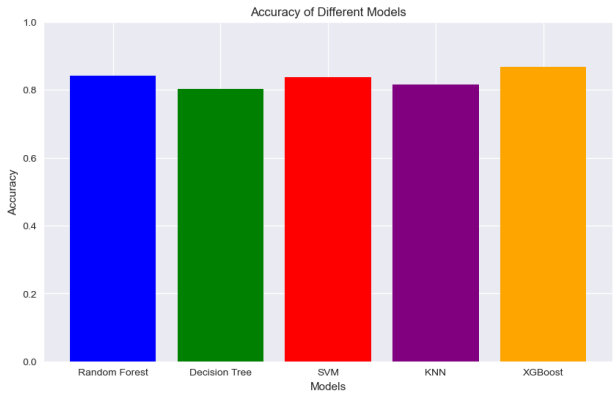
計算各個數值變量之間的相關性，特別是與收入水平的相關性。使用皮爾森相關係數來衡量變量之間的線性關係。

結果如下：

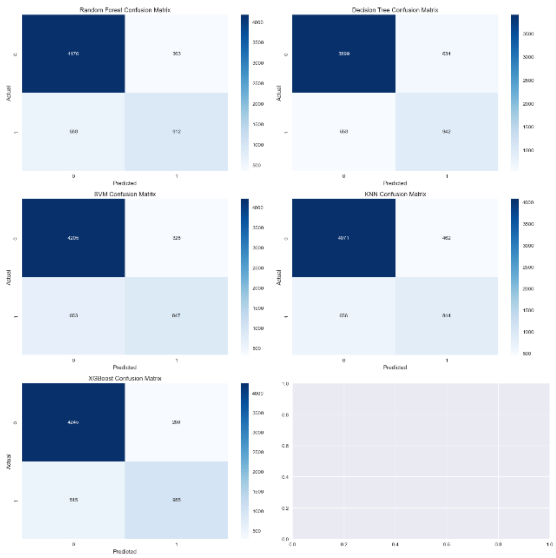
1. 教育年數與收入有較高的正相關性，表明受教育程度越高，收入越有可能超過 50K。
2. 每週工作小時數與收入也有一定的正相關性，表明工作時間越長，收入越高的可能性越大。
3. 年齡與收入有中等程度的相關性。

機器學習預測

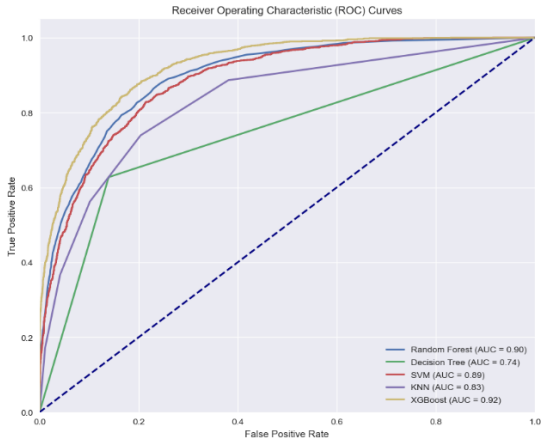
隨機森林和 XGBoost 模型在準確率和穩定性方面表現較好



準確率



混淆矩陣



ROC 曲線

五. 結論:整體的收穫及後續可能的延伸方向

本次試著使用 Python 撰寫程式碼透過以上方法，建構多個收入預測模型，並比較模型的性能。隨機森林和 XGBoost 模型在準確率和穩定性方面表現較好，在相關性方面，教育年數與收入有較高的正相關性，每週工作小時數與收入也有一定的正相關性。

六. 參考資料

Income Predictor Dataset- US Adult Predict whether income exceeds $50K/yr based on census data : https://www.kaggle.com/datasets/jainaru/adult-income-census-dataset/code

七、程式碼

In [99]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

In [100…
```
plt.style.use('bmh')
```

In [101…
```
target_var = 'income'
```

In [102…
```
data = pd.read_csv('adult.csv')
data
```

Out[102]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family
2	66	?	186061	Some-college	10	Widowed	?	Unmarried
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child
…	…	…	…	…	…	…	…	…
32556	22	Private	310152	Some-college	10	Never-married	Protective-serv	Not-in-family
32557	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife
32558	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband
32559	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried
32560	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child

32561 rows × 15 columns

In [103…
```
import pandas as pd

# 讀取 CSV 文件
file_path = 'adult.csv'
df = pd.read_csv(file_path)
```

```
# 查看數據的基本信息
df_info = df.info()

# 顯示前幾行數據
df_head = df.head()

df_info, df_head
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64 
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64 
 11  capital.loss    32561 non-null  int64 
 12  hours.per.week  32561 non-null  int64 
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

Out[103]:

(None,

	age	workclass	fnlwgt	education	education.num	marital.status	\
0	90	?	77053	HS-grad	9	Widowed	
1	82	Private	132870	HS-grad	9	Widowed	
2	66	?	186061	Some-college	10	Widowed	
3	54	Private	140359	7th-8th	4	Divorced	
4	41	Private	264663	Some-college	10	Separated	

	occupation	relationship	race	sex	capital.gain	\
0	?	Not-in-family	White	Female	0	
1	Exec-managerial	Not-in-family	White	Female	0	
2	?	Unmarried	Black	Female	0	
3	Machine-op-inspct	Unmarried	White	Female	0	
4	Prof-specialty	Own-child	White	Female	0	

	capital.loss	hours.per.week	native.country	income
0	4356	40	United-States	<=50K
1	4356	18	United-States	<=50K
2	4356	40	United-States	<=50K
3	3900	40	United-States	<=50K
4	3900	40	United-States	<=50K

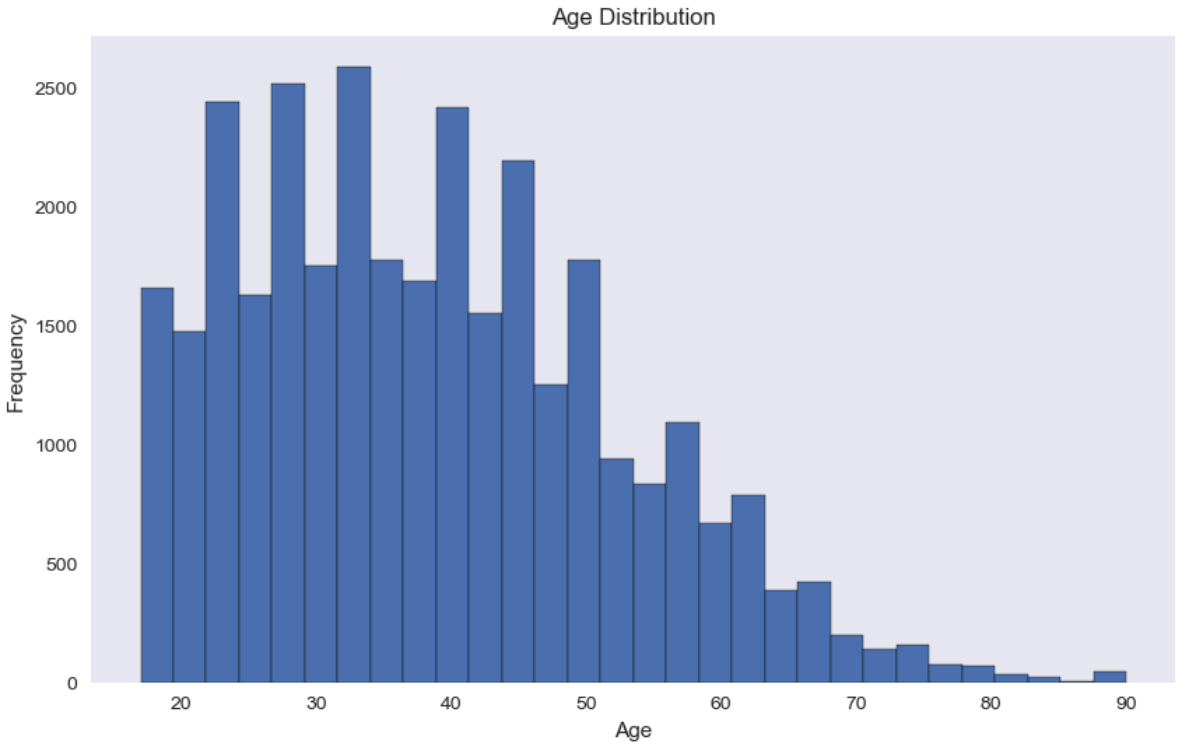
)

In [104…

```
import matplotlib.pyplot as plt

# 設置圖表風格
plt.style.use('seaborn')

# 年齡分佈直方圖
plt.figure(figsize=(10, 6))
df['age'].hist(bins=30, edgecolor='black')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
```

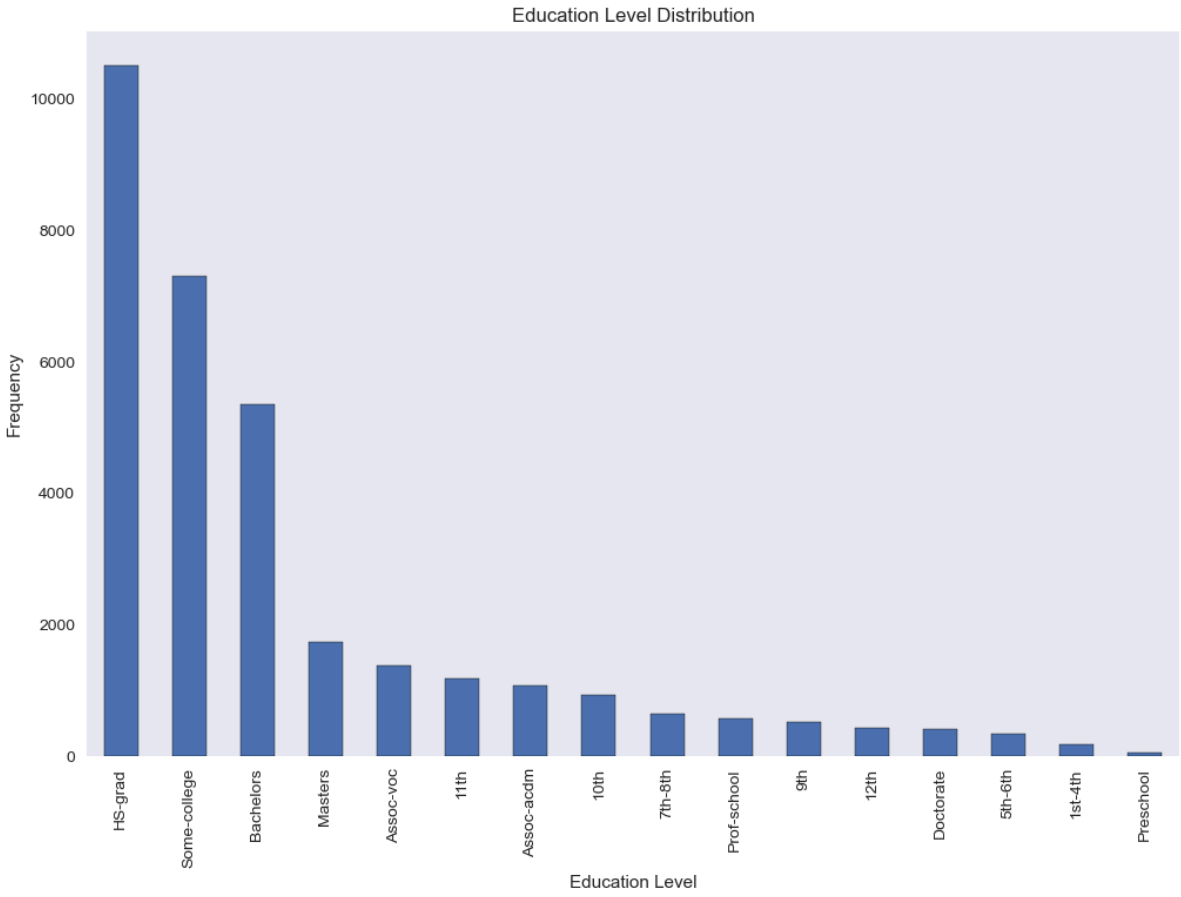
```
plt.grid(False)
plt.show()
```

```
C:\Users\yuchu\AppData\Local\Temp\ipykernel_16572\1581912025.py:4: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn. However, they will remain available as 'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn API instead.
  plt.style.use('seaborn')
```



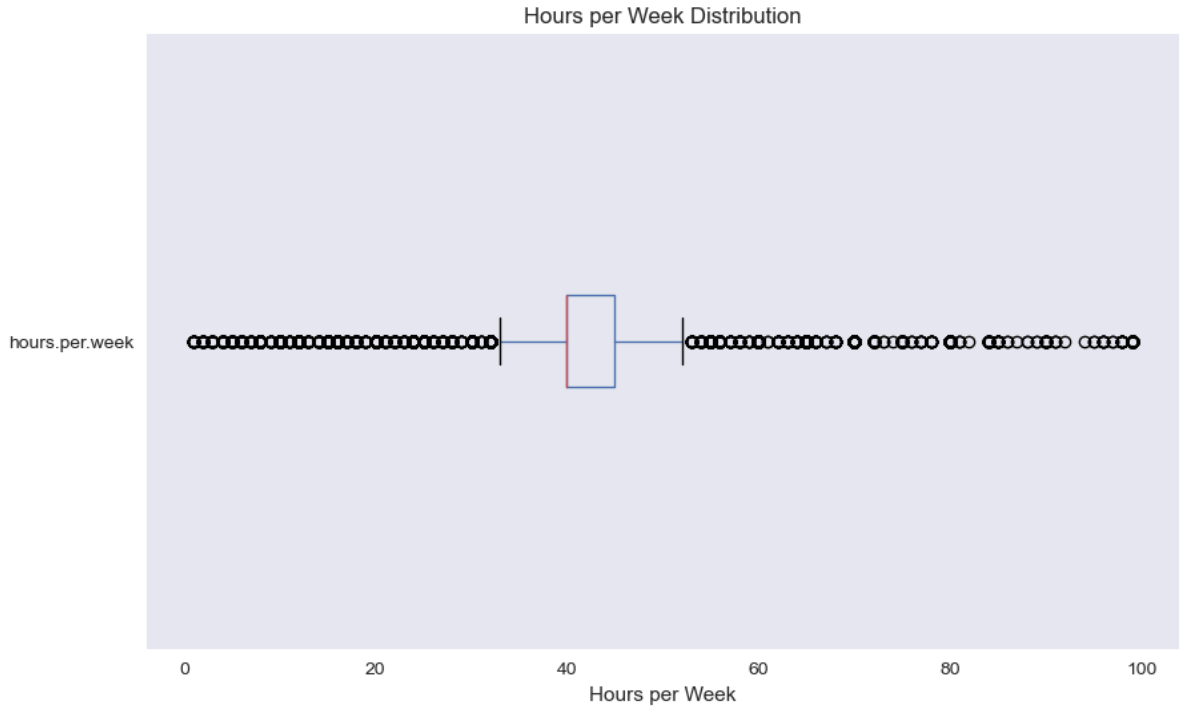
In [105…

```
# 教育程度分佈條形圖
plt.figure(figsize=(12, 8))
df['education'].value_counts().plot(kind='bar', edgecolor='black')
plt.title('Education Level Distribution')
plt.xlabel('Education Level')
plt.ylabel('Frequency')
plt.grid(False)
plt.show()
```



In [106…

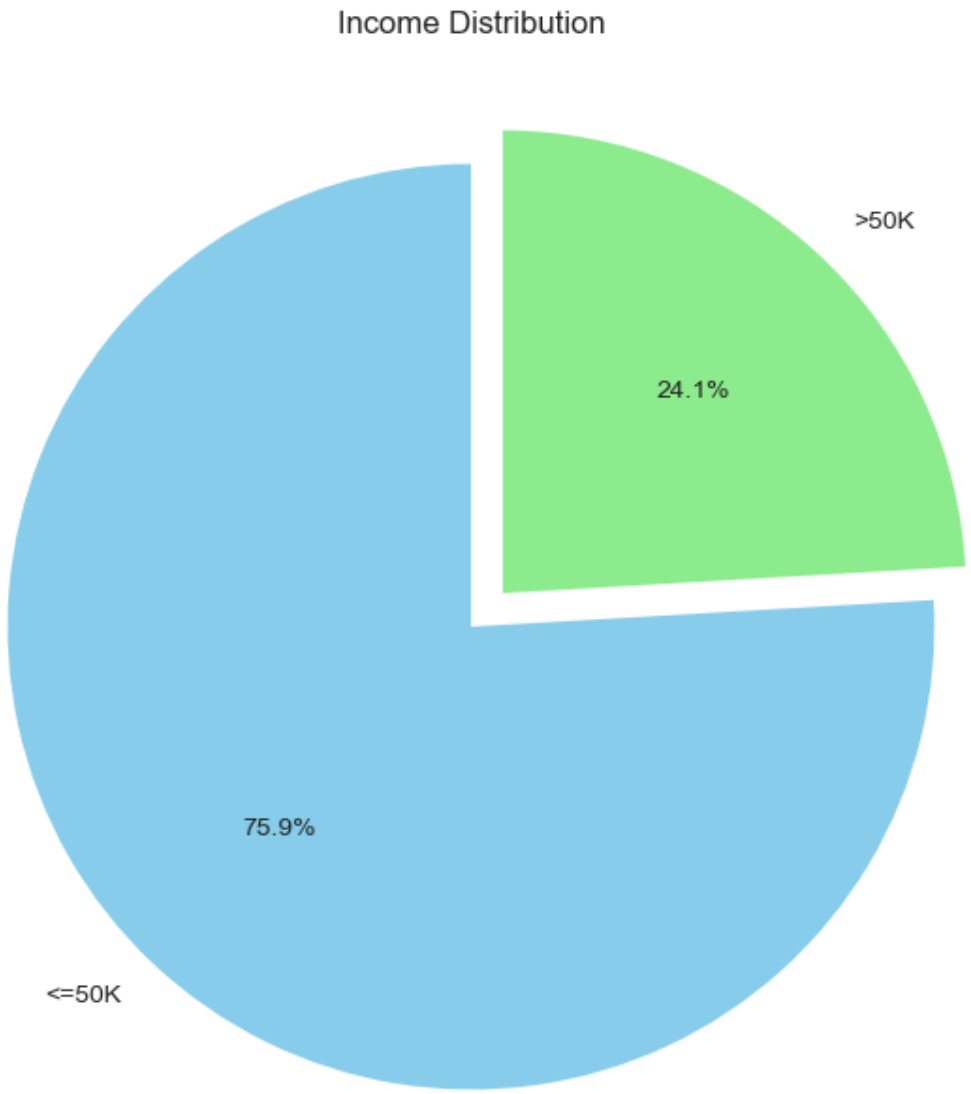
```
# 每周工作小時數的盒鬚圖
plt.figure(figsize=(10, 6))
df.boxplot(column='hours.per.week', vert=False)
plt.title('Hours per Week Distribution')
plt.xlabel('Hours per Week')
plt.grid(False)
plt.show()
```



In [107…

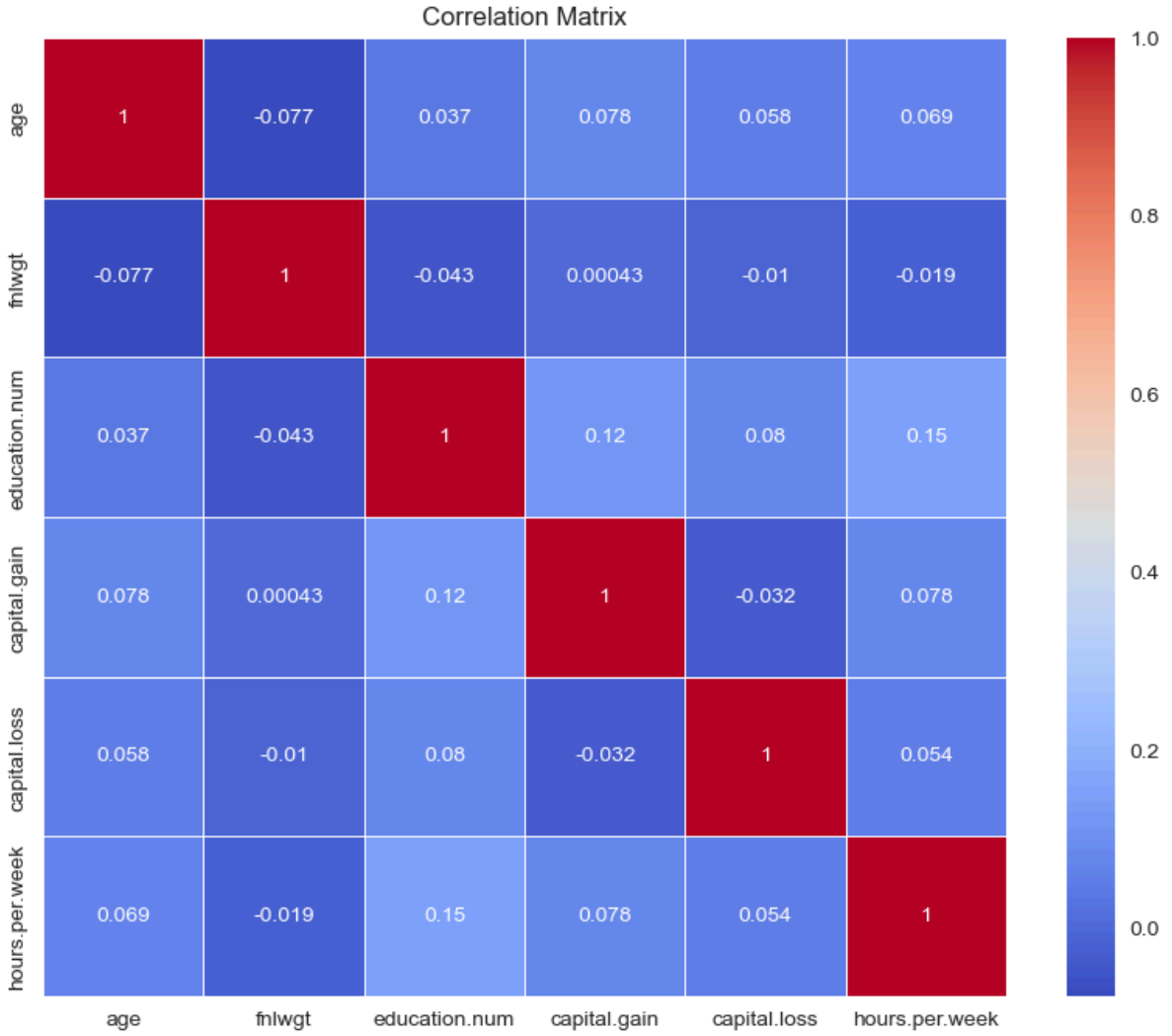
```
# 收入分佈圓餅圖
plt.figure(figsize=(8, 8))
df['income'].value_counts().plot(kind='pie', autopct='%1.1f%%', startangle=90, colo
plt.title('Income Distribution')
```

```
plt.ylabel('')
plt.show()
```



In [108…

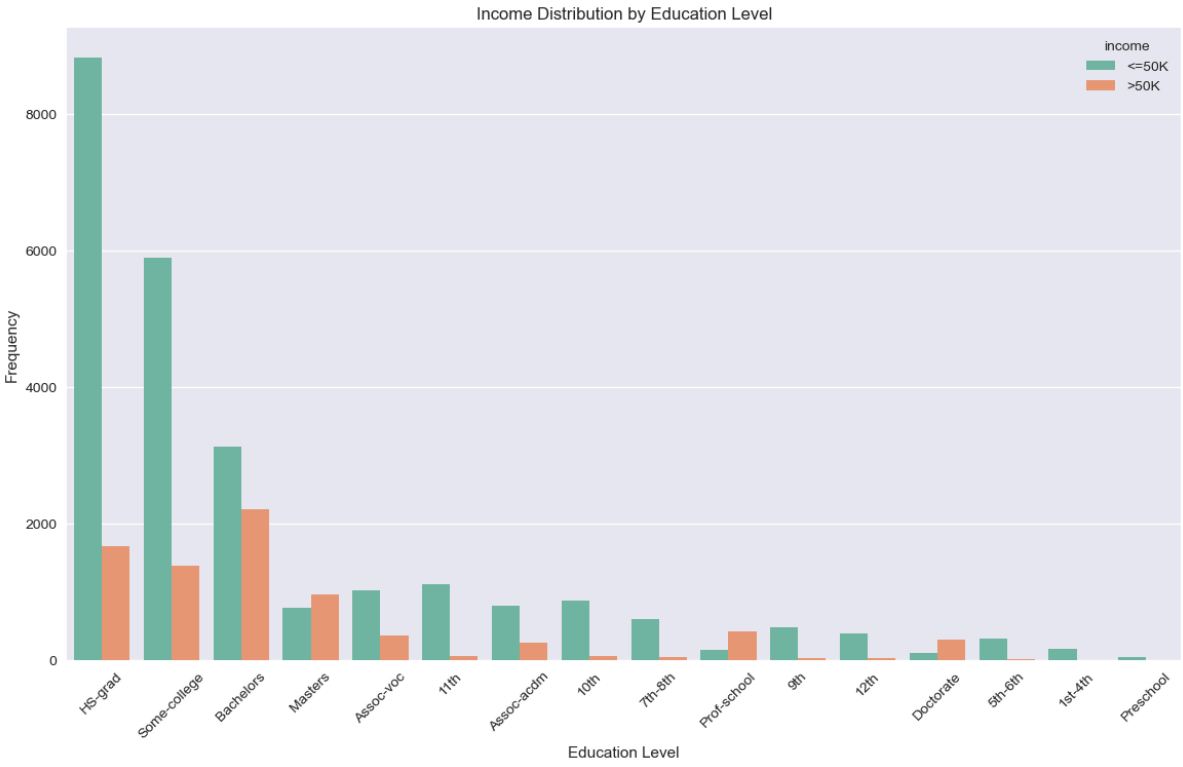
```
import seaborn as sns

# 計算相關性矩陣
correlation_matrix = df[['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital

# 繪製相關性熱力圖
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```

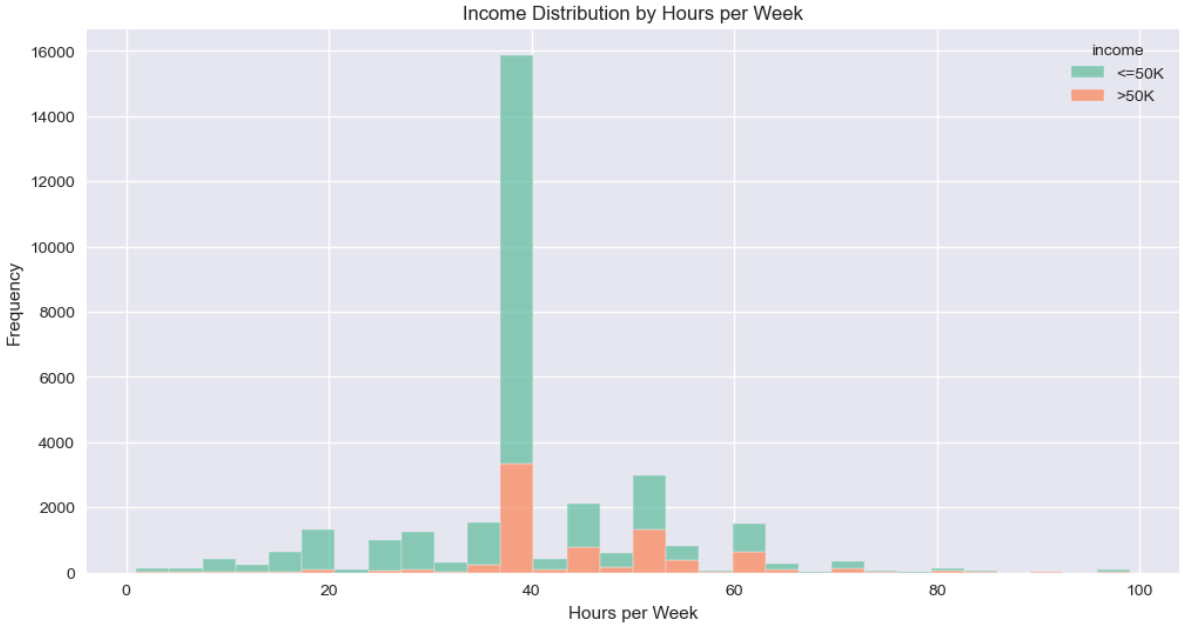


In [109…

```
# 將數據按收入進行分組
df['income'] = df['income'].str.strip()
grouped_by_income = df.groupby('income')

# 1. 收入與年齡的關係
plt.figure(figsize=(12, 6))
sns.histplot(data=df, x='age', hue='income', multiple='stack', bins=30, palette='Se
plt.title('Income Distribution by Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()

# 2. 收入與教育程度的關係
plt.figure(figsize=(14, 8))
sns.countplot(data=df, x='education', hue='income', palette='Set2', order=df['educa
plt.title('Income Distribution by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.show()

# 3. 收入與每周工作小時數的關係
plt.figure(figsize=(12, 6))
sns.histplot(data=df, x='hours.per.week', hue='income', multiple='stack', bins=30,
plt.title('Income Distribution by Hours per Week')
plt.xlabel('Hours per Week')
plt.ylabel('Frequency')
plt.show()
```


Income Distribution by Age
income
<=50K
>50K
Frequency
Age
0
500
1000
1500
2000
2500
20
30
40
50
60
70
80
90


Income Distribution by Education Level
income
<=50K
>50K
Frequency
Education Level
0
2000
4000
6000
8000
HS-grad
Some-college
Bachelors
Masters
Assoc-voc
11th
Assoc-acdm
10th
7th-8th
Prof-school
9th
12th
Doctorate
5th-6th
1st-4th
Preschool


Income Distribution by Hours per Week
income
<=50K
>50K
Frequency
Hours per Week
0
2000
4000
6000
8000
10000
12000
14000
16000
0
20
40
60
80
100

In [110…

```
# 查看缺失值情況
missing_values = df.isnull().sum()

# 處理缺失值（移除包含缺失值的行）
df_cleaned = df.replace(' ?', pd.NA).dropna()

# 確認缺失值已被移除
missing_values_after_cleaning = df_cleaned.isnull().sum()

missing_values, missing_values_after_cleaning
```

Out[110]:

```
(age               0
 workclass         0
 fnlwgt            0
 education         0
 education.num     0
 marital.status    0
 occupation        0
 relationship      0
 race              0
 sex               0
 capital.gain      0
 capital.loss      0
 hours.per.week    0
 native.country    0
 income            0
 dtype: int64,
 age               0
 workclass         0
 fnlwgt            0
 education         0
 education.num     0
 marital.status    0
 occupation        0
 relationship      0
 race              0
 sex               0
 capital.gain      0
 capital.loss      0
 hours.per.week    0
 native.country    0
 income            0
 dtype: int64)
```

In [111…

```
df_cleaned = df.replace('?', pd.NA).dropna()
df_cleaned = df_cleaned.applymap(lambda x: x.strip() if isinstance(x, str) else x)

df_cleaned
```

Out[111]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried
...	...	...	...	...	...	...	...	...
32556	22	Private	310152	Some-college	10	Never-married	Protective-serv	Not-in-family
32557	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife
32558	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband
32559	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried
32560	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child

30162 rows × 15 columns

In [117…

```
df_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 30162 entries, 1 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             30162 non-null  int64 
 1   workclass       30162 non-null  object
 2   fnlwgt          30162 non-null  int64 
 3   education       30162 non-null  object
 4   education.num   30162 non-null  int64 
 5   marital.status  30162 non-null  object
 6   occupation      30162 non-null  object
 7   relationship    30162 non-null  object
 8   race            30162 non-null  object
 9   sex             30162 non-null  object
 10  capital.gain    30162 non-null  int64 
 11  capital.loss    30162 non-null  int64 
 12  hours.per.week  30162 non-null  int64 
 13  native.country  30162 non-null  object
 14  income          30162 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [112…

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
# 編碼分類變量
df_encoded = pd.get_dummies(df_cleaned, columns=['workclass', 'education', 'marital

# 分離特徵和目標變量
X = df_encoded.drop('income', axis=1)
y = df_encoded['income'].apply(lambda x: 1 if x == '>50K' else 0)

# 特徵縮放
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 拆分數據集
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, ran
X
```

Out[112]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclass_Local-gov	w
1	82	132870	9	0	4356	18	False	
3	54	140359	4	0	3900	40	False	
4	41	264663	10	0	3900	40	False	
5	34	216864	9	0	3770	45	False	
6	38	150601	6	0	3770	40	False	
...	...	...	...	...	...	...	...	
32556	22	310152	10	0	0	40	False	
32557	27	257302	12	0	0	38	False	
32558	40	154374	9	0	0	40	False	
32559	58	151910	9	0	0	40	False	
32560	22	201490	9	0	0	20	False	

30162 rows × 96 columns

In [113…

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
# 訓練和評估模型
models = {
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "KNN": KNeighborsClassifier(),
    "XGBoost": XGBClassifier(random_state=42)
}

accuracies = {}
confusion_matrices = {}
roc_curves = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_pred_proba = model.predict_proba(X_test)[:, 1]

    accuracies[name] = accuracy_score(y_test, y_pred)
    confusion_matrices[name] = confusion_matrix(y_test, y_pred)
    fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
```

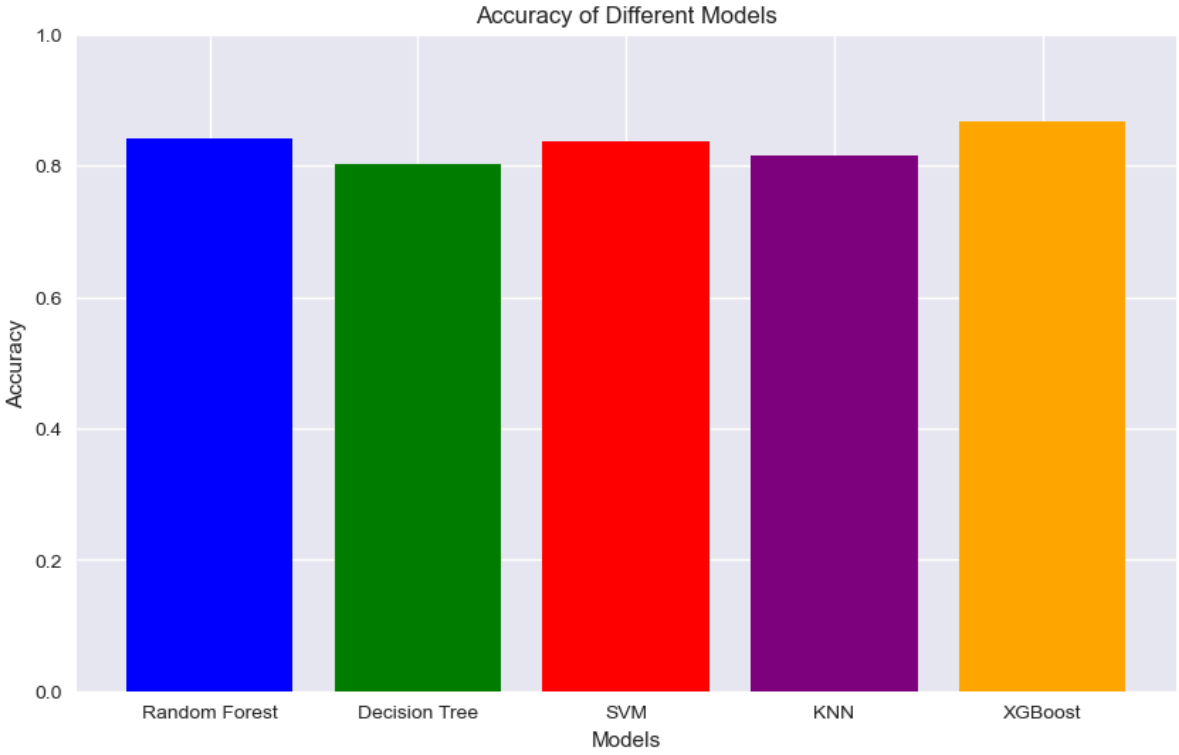
```
    roc_curves[name] = (fpr, tpr, auc(fpr, tpr))

    print(f"{name} Accuracy: {accuracies[name]}")
    print(f"{name} Confusion Matrix:\n{confusion_matrices[name]}")
    print(f"{name} Classification Report:\n{classification_report(y_test, y_pred)}\
```

```
Random Forest Accuracy: 0.8423669816011934
Random Forest Confusion Matrix:
[[4170  363]
 [ 588  912]]
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.92      0.90      4533
           1       0.72      0.61      0.66      1500

    accuracy                           0.84      6033
   macro avg       0.80      0.76      0.78      6033
weighted avg       0.84      0.84      0.84      6033


Decision Tree Accuracy: 0.8024200232057019
Decision Tree Confusion Matrix:
[[3899  634]
 [ 558  942]]
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.86      0.87      4533
           1       0.60      0.63      0.61      1500

    accuracy                           0.80      6033
   macro avg       0.74      0.74      0.74      6033
weighted avg       0.81      0.80      0.80      6033


SVM Accuracy: 0.8373943311785181
SVM Confusion Matrix:
[[4205  328]
 [ 653  847]]
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.93      0.90      4533
           1       0.72      0.56      0.63      1500

    accuracy                           0.84      6033
   macro avg       0.79      0.75      0.76      6033
weighted avg       0.83      0.84      0.83      6033


KNN Accuracy: 0.814685894248301
KNN Confusion Matrix:
[[4071  462]
 [ 656  844]]
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.90      0.88      4533
           1       0.65      0.56      0.60      1500

    accuracy                           0.81      6033
   macro avg       0.75      0.73      0.74      6033
weighted avg       0.81      0.81      0.81      6033


XGBoost Accuracy: 0.8668987236863915
XGBoost Confusion Matrix:
[[4245  288]
 [ 515  985]]
```

```
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.94      0.91      4533
           1       0.77      0.66      0.71      1500

    accuracy                           0.87      6033
   macro avg       0.83      0.80      0.81      6033
weighted avg       0.86      0.87      0.86      6033
```

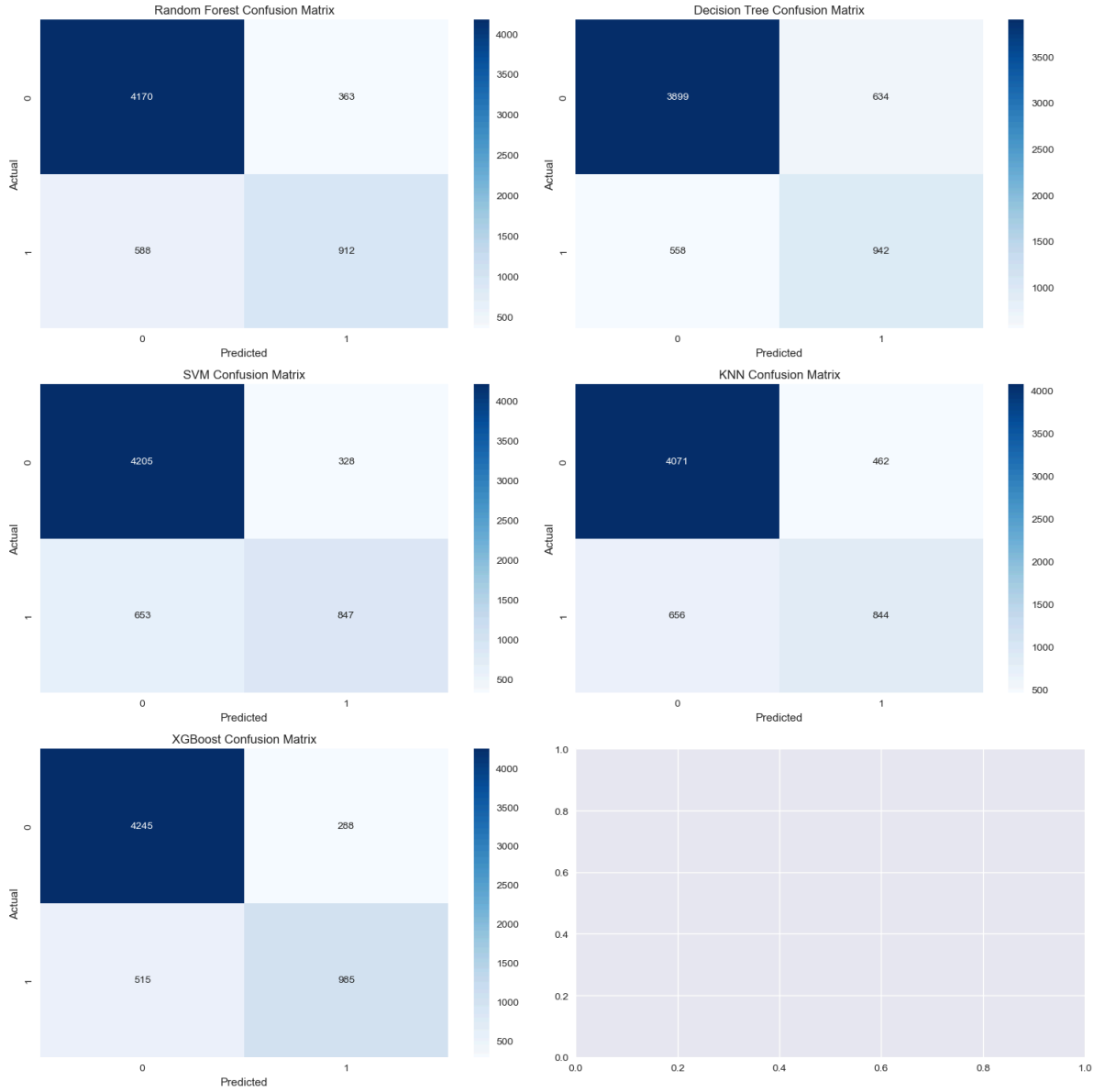
In [114…

```python
# 繪製精度圖表
plt.figure(figsize=(10, 6))
plt.bar(accuracies.keys(), accuracies.values(), color=['blue', 'green', 'red', 'pur
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy of Different Models')
plt.ylim(0, 1)
plt.savefig('model_accuracies.png')
plt.show()
```



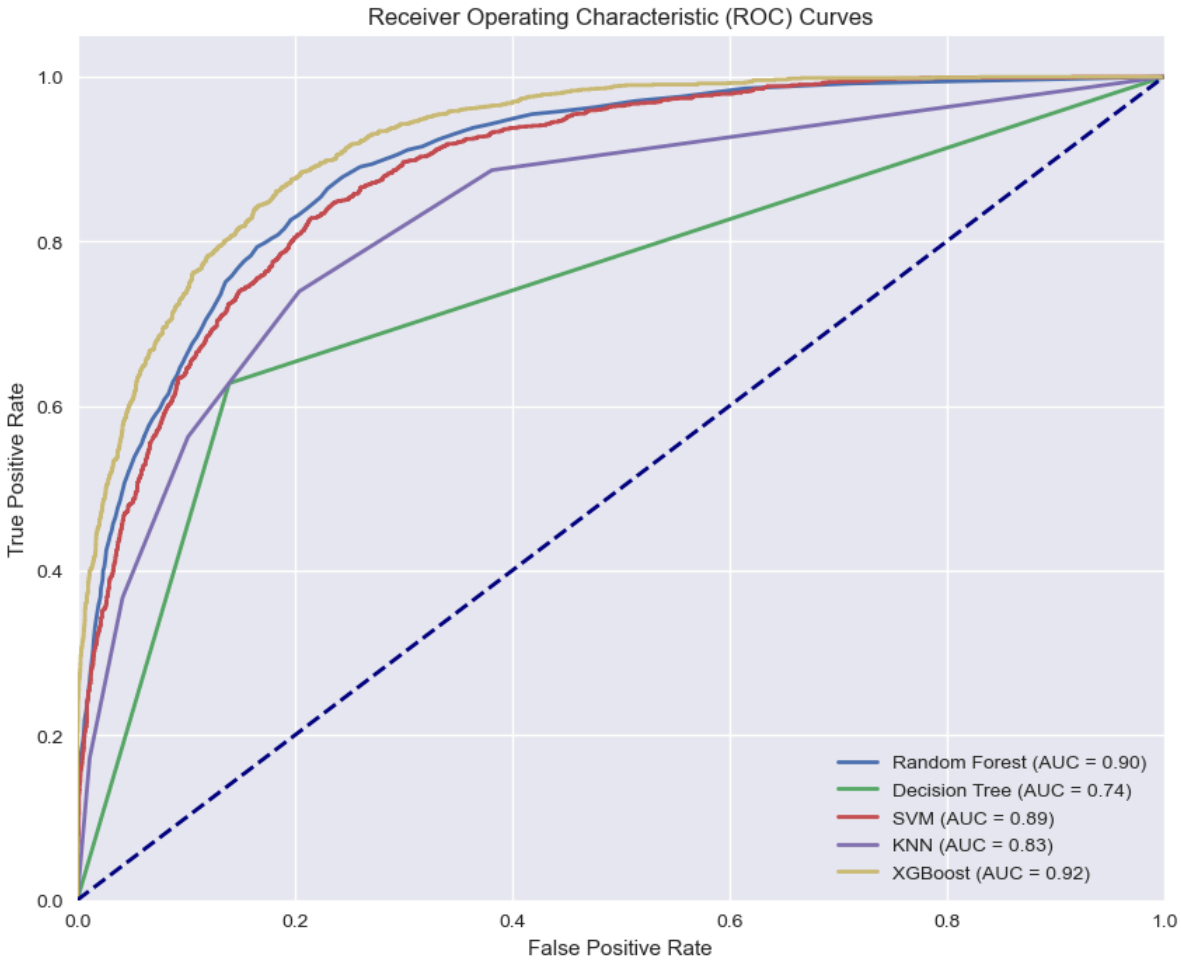
In [115…

```python
# 繪製混淆矩陣
fig, axes = plt.subplots(3, 2, figsize=(15, 15))
axes = axes.ravel()

for i, (name, cm) in enumerate(confusion_matrices.items()):
    sns.heatmap(cm, annot=True, fmt='d', ax=axes[i], cmap='Blues')
    axes[i].set_title(f'{name} Confusion Matrix')
    axes[i].set_xlabel('Predicted')
    axes[i].set_ylabel('Actual')

plt.tight_layout()
plt.savefig('confusion_matrices.png')
plt.show()
```



In [116…

```python
# 繪製 ROC 曲線
plt.figure(figsize=(10, 8))

for name, (fpr, tpr, roc_auc) in roc_curves.items():
    plt.plot(fpr, tpr, lw=2, label=f'{name} (AUC = {roc_auc:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curves')
plt.legend(loc='lower right')
plt.savefig('roc_curves.png')
plt.show()
```


Receiver Operating Characteristic (ROC) Curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Random Forest (AUC = 0.90)
Decision Tree (AUC = 0.74)
SVM (AUC = 0.89)
KNN (AUC = 0.83)
XGBoost (AUC = 0.92)