查看資料內容

將資料讀入之後，用head()查看前幾筆資料的欄位內容。

```
data_df.head()
```

0.6s Python

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
				Allen, Mr. William								

查看資料相關性

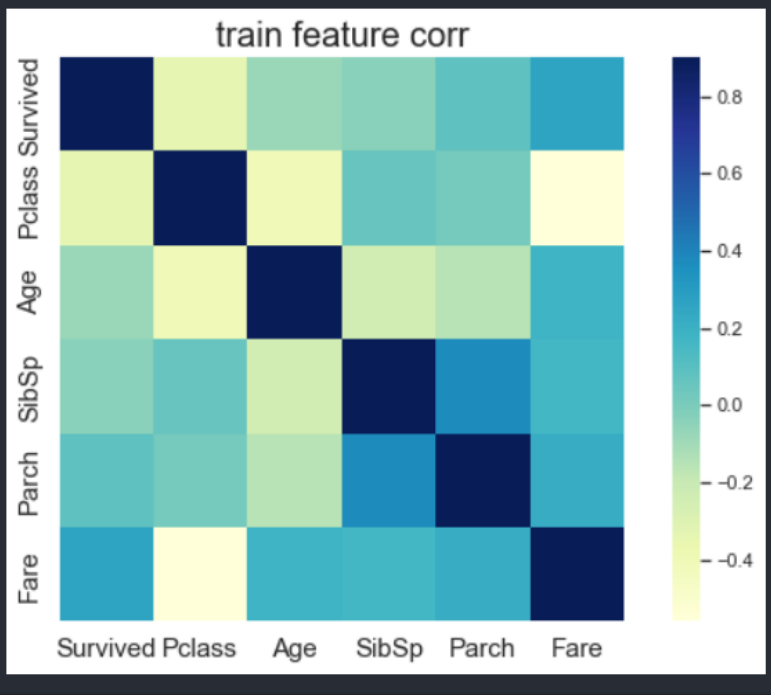
數值資料

算出correlation後，使用seaborn畫heatmap，並將顏色調為相關性越高顏色越深，方便觀察。

```
sns.set(context="paper", font="monospace")
sns.set(style="white")
f, ax = plt.subplots(figsize=(10,6))
data_corr = data_df.drop('PassengerId',axis=1).corr()
sns.heatmap(data_corr, ax=ax, vmax=.9, square=True,cmap='YlGnBu')
ax.set_xticklabels(data_corr.index, size=15)
ax.set_yticklabels(data_corr.columns[::1], size=15)
ax.set_title('train feature corr', fontsize=20)
```

✓ 0.2s

```
Text(0.5, 1.0, 'train feature corr')
```



因為我們最後預測的目標是survived，所以以它為主要觀察的對象。
可以看到survived最與fare呈正相關，而與Pclass呈負相關。

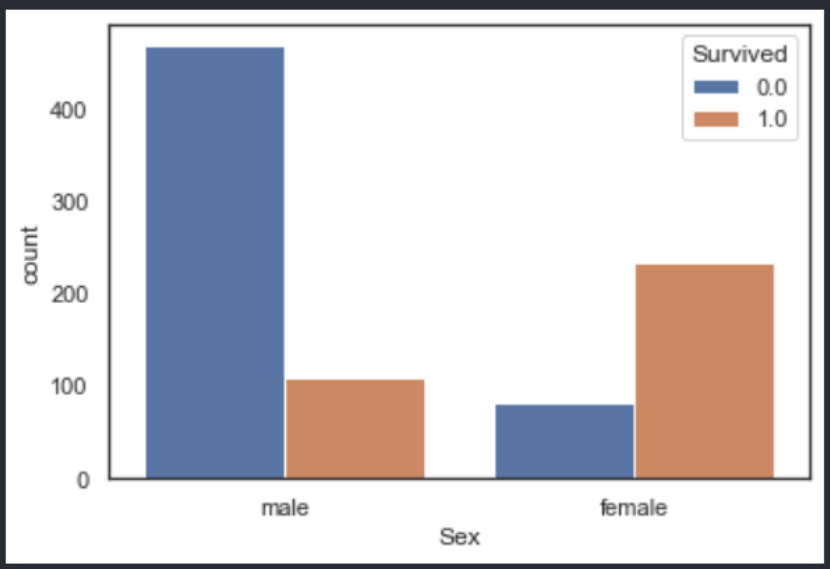
非數值資料

製作性別的長條圖

```
sns.countplot(df_data['Sex'], hue=df_data['Survived'])
```

0.2s

```
<AxesSubplot:xlabel='Sex', ylabel='count'>
```



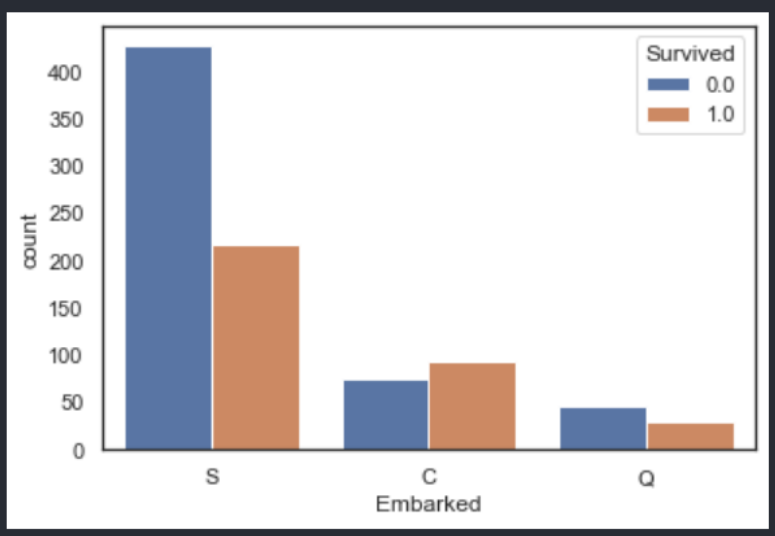
可以發現姓別對生存率有很大的關聯。

以登船地點做長條圖:

```
sns.countplot(df_data['Embarked'], hue=df_data['Survived'])
```

0.2s

```
<AxesSubplot:xlabel='Embarked', ylabel='count'>
```



可以觀察到S的死亡率高，而C存活率高。

資料前處理

feature creating

因為parch跟sibsp都是關於家庭成員的人數，且都略有跟survived有相關，於是將其合併為家庭成員人數，來提升相關性。

```
df_data['Family'] = df_data['SibSp'] + df_data['Parch'] + 1
```

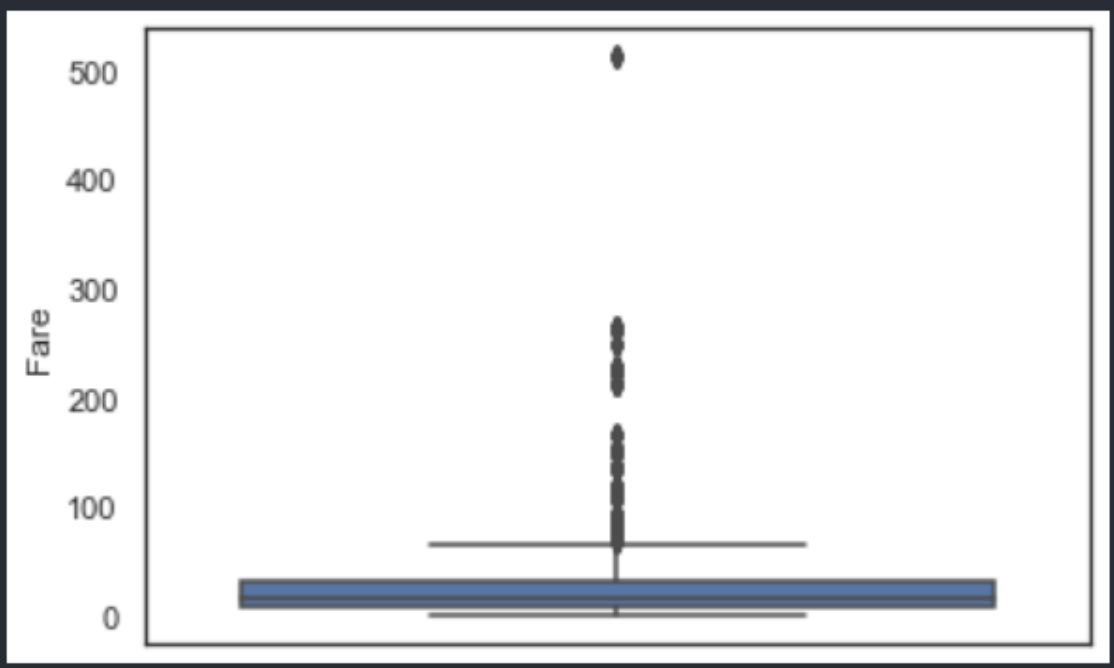
rescale

因為Fare的分佈太過懸殊，我們將它rescale，讓算法可以穩定。

```
sns.boxplot(y=df_data['Fare'])
```

0.4s

```
<AxesSubplot:ylabel='Fare'>
```




```
scaler = preprocessing.StandardScaler()
scaler = scaler.fit(df_data['Fare'].values.reshape(-1, 1))
df_data['Fare'] = scaler.transform(df_data['Fare'].values.reshape(-1, 1))
```

0.4s Python

填補缺失值

用info()看各欄位是否有缺漏值。

```
data_df.info()
```

[17] ✓ 0.4s

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64
 3   Name         1309 non-null   object
 4   Sex          1309 non-null   object
 5   Age          1046 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   int64
 8   Ticket       1309 non-null   object
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object
 11  Embarked     1307 non-null   object
```

因為在上面我們已經做過關聯的分析，我們只需去填補我們需要的欄位。
Fare缺失值以中位數補上:

```
df_data['Fare'] = df_data['Fare'].fillna(df_data['Fare'].median())
```

✓ 0.4s

Embarked以最多人登入的S補上。

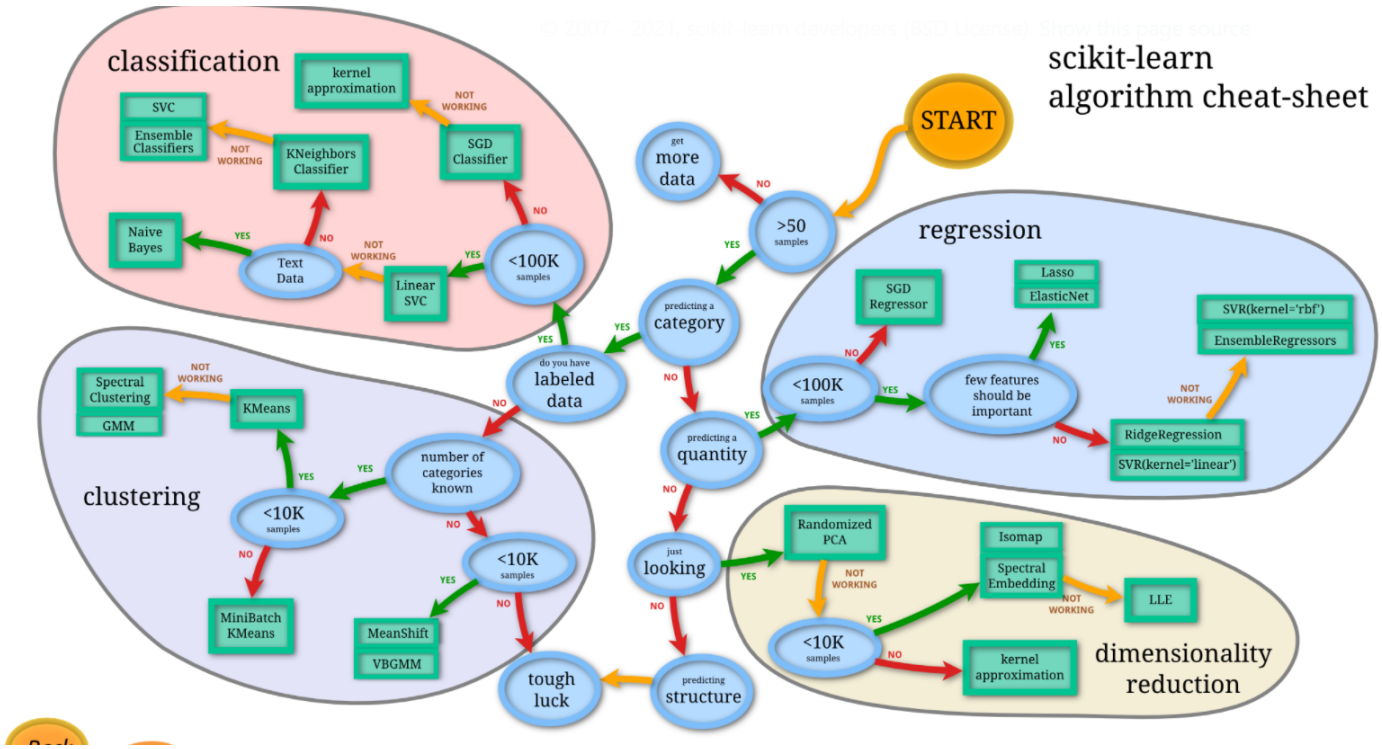
```
df_data['Embarked'] = df_data['Embarked'].fillna('S')
```

0.7s

因為age和cabin缺失太多，我認為隨意的補值只會降低準確率，於是我們不會考慮使用它們作為訓練。

選擇算法

本資料集的目標為預測是否生存，也就是classifiction。
參考sklearn上的算法選擇的路線圖，考慮落在classification的算法，加上網路上的文章許多都採用random forest取得不錯的預測結果，於是採用random forest進行預測。



採取預設的gini值，因為發現使用entropy並不會特別提高表現。

```
select_feat = ['Sex','Pclass','Embarked','Family']
selector = RandomForestClassifier(n_estimators=250,criterion='gini',min_samples_split=20)
selector.fit(X[select_feat], Y)
print(selector)
```

✓ 0.3s

```
RandomForestClassifier(min_samples_split=20, n_estimators=250)
```

Name	Submitted	Wait time	Execution time	Score
submit.csv	just now	1 seconds	0 seconds	0.77033

Complete

Jump to your position on the leaderboard

結果

7955	TsungHanChou		0.77511	3	12m

(其最高的score為把random forest再套用sklearn的feature selection的結果)