

Close

Alaa Waleed Ali Humaid
ID:261002982
Practical ML assignmet

Alaa Waleed Ali Humaid ID:261002982 Practical ML
assignmet

```
!pip install panda
!pip install numpy
!pip install scikit-learn
```

```
Requirement already satisfied: panda in /usr/local/lib/python3.12/dist-packages (0.3.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from panda) (75.2.0)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from panda) (2.32.4)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests) (3.10.1)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests) (2025.11.12)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.5.0)
```

```
#import laibraries
```

```
import pandas as pd
import numpy as np
```

```
df= pd.read_csv('train.csv')
print(df.isnull().sum())
```

```
PassengerId      0
Survived          0
Pclass           0
Name             0
Sex              0
Age             177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin           687
Embarked         2
dtype: int64
```

```
# handel numric missing values and clean dataset
```

```
df=df.fillna(df.mean(numeric_only=True))
print(df.isnull().sum())
```

```
PassengerId      0
Survived          0
Pclass           0
Name             0
Sex              0
Age             177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin           687
Embarked         2
dtype: int64
```

```
df=df.drop(['Cabin','Name','Ticket'], axis=1)
print(df.isnull().sum())
```

```
PassengerId    0
Survived        0
Pclass          0
Sex             0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       2
dtype: int64
```

```
# handel categoric missing values and clean dataset
```

```
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
print(df.isnull().sum())
```

```
PassengerId    0
Survived        0
Pclass          0
Sex             0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       0
dtype: int64
```

```
# Conver "sex" column to numric value
```

```
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
df['Embarked'] = df['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})
```

```
print(df.dtypes)
```

```
PassengerId    int64
Survived        int64
Pclass          int64
Sex             int64
Age            float64
SibSp          int64
Parch          int64
Fare           float64
Embarked       int64
dtype: object
```

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

```
# split dataset into training and testing sets
```

```
x= df.drop('Survived', axis=1)
y= df['Survived']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

model= RandomForestClassifier()
```

```
# Apply random forest model with Grid Search to find best parameter
```

```
param_grid = {'n_estimators':[50, 100], 'max_depth': [3,5,10]}

grid= GridSearchCV(model, param_grid, cv=3)
grid.fit(x_train, y_train)
```

```
best_model= grid.best_estimator_

y_pred = best_model.predict(x_test)

print("best Params:", grid.best_params_)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
best Params: {'max_depth': 5, 'n_estimators': 100}
Accuracy: 0.8156424581005587
```

	precision	recall	f1-score	support
0	0.80	0.91	0.85	105
1	0.85	0.68	0.75	74
accuracy			0.82	179
macro avg	0.82	0.79	0.80	179
weighted avg	0.82	0.82	0.81	179

```
# Apply feature selection using select KBest (filter method) to choose best features
```

```
from sklearn.feature_selection import SelectKBest, f_classif
selector= SelectKBest(f_classif, k=5)
x_new= selector.fit_transform(x,y)
```

```
selected_features= x.columns[selector.get_support()]
print(selected_features)
```

```
Index(['Pclass', 'Sex', 'Parch', 'Fare', 'Embarked'], dtype='object')
```

```
# identify important features using features importance from random forest (embedded method)
```

```
import matplotlib.pyplot as plt
model.fit(x,y)
importance= model.feature_importances_
```

```
features= x.columns
```

```
for f, imp in zip(features, importance):
    print(f, imp)
```

```
PassengerId 0.19390226201083965
Pclass 0.08425783972534612
Sex 0.23318839378708903
Age 0.17766226760325818
SibSp 0.043233853909556884
Parch 0.034824248338235254
Fare 0.20259736719994015
Embarked 0.03033376742573473
```

```
#train the model using selected features
```

```
x_train,x_test, y_train, y_test= train_test_split(x_new, y, test_size=0.2, random_state=42)
```

```
model.fit(x_train, y_train)
y_pred= model.predict(x_test)
```

```
print("New Accuracy:")
accuracy_score(y_test, y_pred)
```

```
New Accuracy:
0.8156424581005587
```

Accuracy remained the same that means the selected features are sufficient for prediction and removing less important features didn't negatively affect the model performance.

