MINI PROJECT : Build a machine learning model that predicts the type of people who survived the Titanic shipwreck using passenger data (i.e. name, age, gender, socio-economic class, etc.). Dataset Link: https://www.kaggle.com/competitions/titanic/data

```
# 1. Import Libraries
# ------------------------------
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
```

```
# 2. Load Data
# ------------------------------
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')


print("Train dataset shape:", train.shape)
print("Test dataset shape:", test.shape)
```

```
Train dataset shape: (891, 12)
Test dataset shape: (418, 11)
```

```
# 3. Explore Data
# ------------------------------
train.info()
train.head()

# Correlation heatmap (numeric columns only)
train_numeric = train.select_dtypes(include=['int64', 'float64'])
train_numeric.corr().style.background_gradient(cmap='BuGn')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

```
# 4. Drop Unnecessary Columns
# ------------------------------
train.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)
test.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)
```

```
# 5. Check Missing Values
# ------------------------------
train.isna().sum()
test.isna().sum()
```

	0
Pclass	0
Sex	0
Age	86
SibSp	0
Parch	0
Fare	1
Embarked	0

dtype: int64

```
# 6. Fill Missing Values
# ------------------------------
train['Embarked'] = train['Embarked'].fillna(train['Embarked'].dropna().mode()[0])
test['Fare'] = test['Fare'].fillna(test['Fare'].dropna().mean())
```

```
# 7. Feature Engineering (Sex & Age)
# ------------------------------
guess_ages = np.zeros((2,3))
combine = [train, test]

# Convert Sex to numeric
for ds in combine:
    ds['Sex'] = ds['Sex'].map({'female':1, 'male':0}).astype(int)

# Fill missing Age
for ds in combine:
    for i in range(2):
        for j in range(3):
            guess_df = ds[(ds['Sex']==i) & (ds['Pclass']==j+1)]['Age'].dropna()
            age_guess = guess_df.median()
            guess_ages[i,j] = int(age_guess/0.5 + 0.5) * 0.5
    for i in range(2):
        for j in range(3):
            ds.loc[(ds['Age'].isnull()) & (ds['Sex']==i) & (ds['Pclass']==j+1), 'Age'] = guess_ages[i,j]
    ds['Age'] = ds['Age'].astype(int)
```

```
# 8. Cleaned Dataset Preview (Sir's Format)
# ------------------------------
train_cleaned = train[['Survived','Pclass','Sex','Age','SibSp','Parch','Fare','Embarked']]
train_cleaned.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22	1	0	7.2500	S
1	1	1	1	38	1	0	71.2833	C
2	1	3	1	26	0	0	7.9250	S
3	1	1	1	35	1	0	53.1000	S
4	0	3	0	35	0	0	8.0500	S

Next steps: Generate code with `train_cleaned` New interactive sheet

```
# 9. Prepare Data for Modeling
# ------------------------------
X_train = pd.get_dummies(train.drop(['Survived'], axis=1))
X_test = pd.get_dummies(test)
y_train = train['Survived']
```

```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
```

```
X_train shape: (891, 9)
X_test shape: (418, 9)
```

```
# 10. Define Function to Print Scores
# ------------------------------
def print_scores(model, X_train, y_train, predictions, cv_splits=10):
    print("Training Accuracy: %.5f" % model.score(X_train, y_train))
    CV_scores = cross_val_score(model, X_train, y_train, cv=cv_splits)
    print("Cross-validation scores:\n", CV_scores)
    print("Minimum CV score: %.3f" % min(CV_scores))
    print("Maximum CV score: %.3f" % max(CV_scores))
    print("Mean CV score: %.5f ± %0.2f" % (CV_scores.mean(), CV_scores.std()*2))
```

```
# 11. Train Random Forest Classifier
# ------------------------------
model = RandomForestClassifier(
    n_estimators=80,
    max_depth=5,
    max_features=8,
    min_samples_split=3,
    random_state=7
)

model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

```
# 12. Evaluate Model
# ------------------------------
print_scores(model, X_train, y_train, predictions)
```

```
Training Accuracy: 0.85859
Cross-validation scores:
 [0.76666667 0.85393258 0.75280899 0.91011236 0.88764045 0.80898876
 0.80898876 0.78651685 0.87640449 0.84269663]
Minimum CV score: 0.753
Maximum CV score: 0.910
Mean CV score: 0.82948 ± 0.10
```