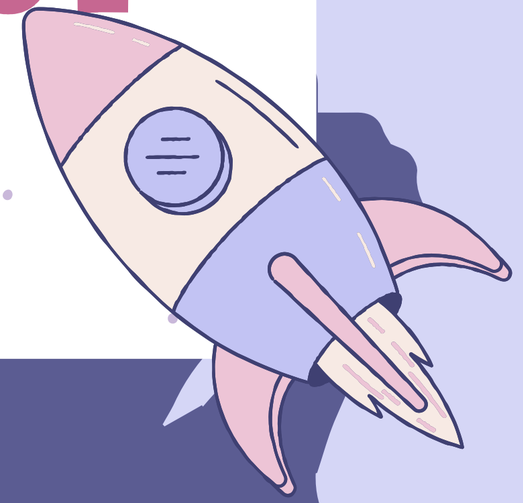
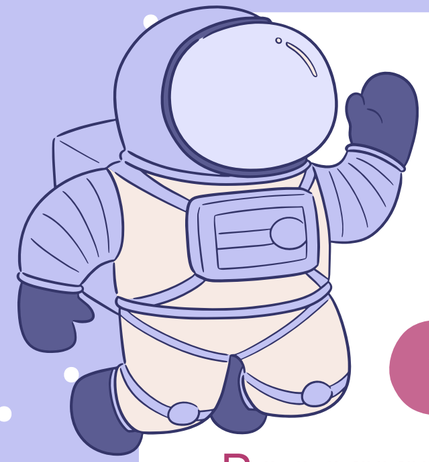
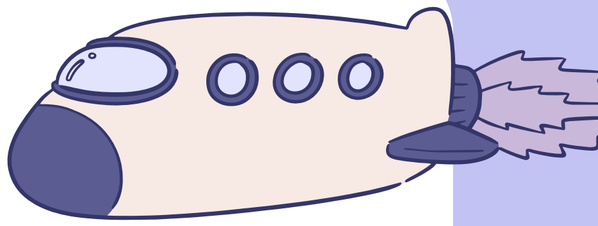
SPACESHIP
TITANIC CHALLENGE

Group.13
110511091 蔡炅宸
110550143 洪巧芸



OUTLINE





1

Recommendation System

The origin of space titanic dataset.

2

Data Preprocessing

Including the process of importing and preparing the data before conducting any analysis.

3

Math of Evaluation

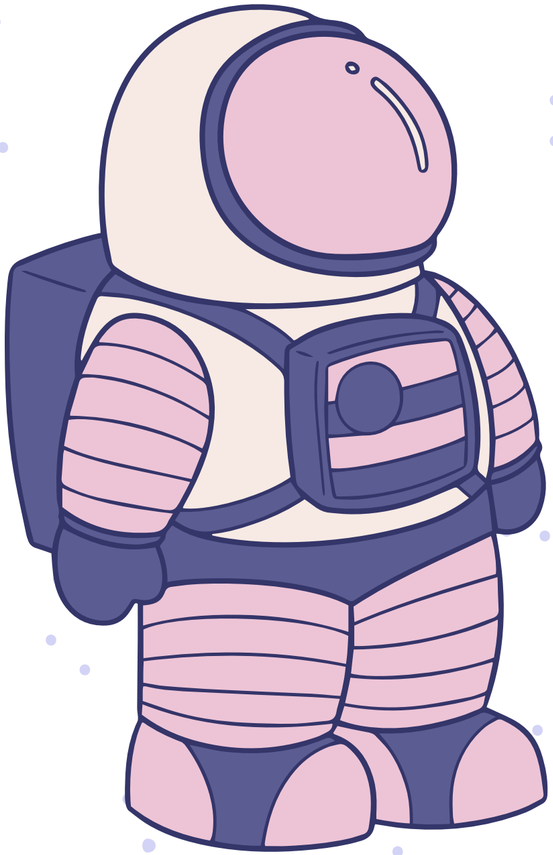
A brief introduction to the algorithm that we used in our code.

4

Evaluation

The comparison of the accuracy of the results trained by decision tree and random forest

Space Titanic DATASET



It starts with a competition launched on Kaggle. This competition is designed as an update to the popular Titanic competition, aiming to assist individuals new to data science in learning the fundamentals of machine learning, becoming familiar with Kaggle's platform, and connecting with others in the community

DATA PREPROCESSING

1 Data importing

```
# Download the file to a local disc
train_download.GetContentFile('train_file.csv')
test_download.GetContentFile('test_file.csv')


# Specify the data type for the problematic column
dtype_dict = {6: 'str'}

train_data = pd.read_csv("train_file.csv", dtype=dtype_dict, low_memory=False)
test_data = pd.read_csv("test_file.csv", dtype=dtype_dict, low_memory=False)
```

DATA PREPROCESSING

2 Data splitting

```
# Function to split the 'Cabin' column into 'Deck', 'Num', and 'Side'
def split_cabin(dataframe):
    # Splitting the 'Cabin' column
    cabin_split = dataframe['Cabin'].str.split('/', expand=True)
    cabin_split.columns = ['Deck', 'Num', 'Side']

    # Concatenating the new columns with the original dataframe
    return pd.concat([dataframe.drop(columns=['Cabin']), cabin_split], axis=1)

# Apply the function to both training and testing datasets
train_df = split_cabin(train_data)
test_df = split_cabin(test_data)
```

DATA PREPROCESSING

2 Data splitting

	PassengerId	HomePlanet	CryoSleep	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Name	Transported	Deck	Num	Side
0	0001_01	Europa	False	TRAPPIST-1e	39.0	False	0.0	0.0	0.0	0.0	0.0	Maham Ofracculy	False	B	0	P
1	0002_01	Earth	False	TRAPPIST-1e	24.0	False	109.0	9.0	25.0	549.0	44.0	Juanna Vines	True	F	0	S
2	0003_01	Europa	False	TRAPPIST-1e	58.0	True	43.0	3576.0	0.0	6715.0	49.0	Altark Susent	False	A	0	S
3	0003_02	Europa	False	TRAPPIST-1e	33.0	False	0.0	1283.0	371.0	3329.0	193.0	Solam Susent	False	A	0	S
4	0004_01	Earth	False	TRAPPIST-1e	16.0	False	303.0	70.0	151.0	565.0	2.0	Willy Santantines	True	F	1	S

DATA PREPROCESSING

3 Data flitering

```
def preprocess_data(train_df, test_df):
    # Identifying categorical and numerical columns
    categorical_cols = ['HomePlanet', 'CryoSleep', 'Destination', 'VIP', 'Deck', 'Side']
    numerical_cols = ['Age', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']

    # Pipeline for imputing and encoding categorical columns
    categorical_transformer = Pipeline(steps=[
        ('imputer', SimpleImputer(strategy='most_frequent')),
        ('onehot', OneHotEncoder(handle_unknown='ignore'))
    ])

    # Pipeline for imputing and scaling numerical columns
    numerical_transformer = Pipeline(steps=[
        ('imputer', SimpleImputer(strategy='median')),
        ('scaler', StandardScaler())
    ])
```

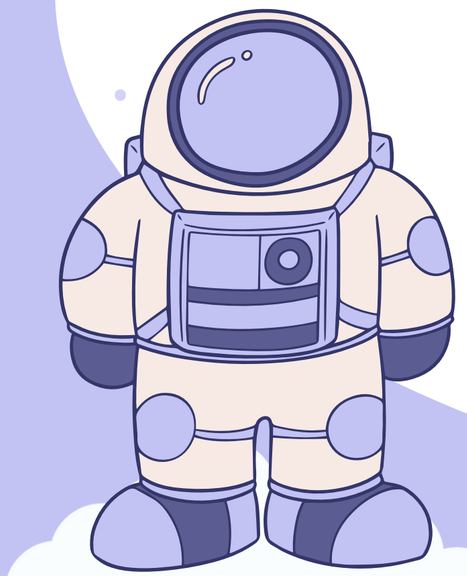
DATA PREPROCESSING

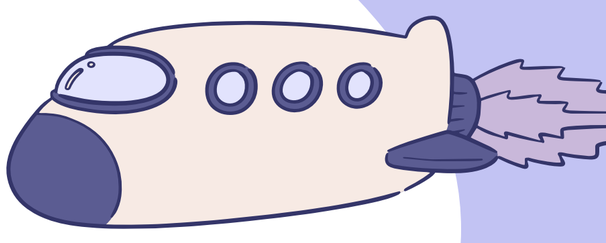
4 Data merging

```
# Combining transformers into a ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ])

# Applying the transformations to the training data
X_train = preprocessor.fit_transform(train_df.drop(['PassengerId', 'Name', 'Transported'], axis=1))
y_train = train_df['Transported']

# Applying the same transformations to the test data (excluding target variable)
X_test = preprocessor.transform(test_df.drop(['PassengerId', 'Name'], axis=1))

# Getting categorical feature names after one-hot encoding
categorical_features = preprocessor.named_transformers_['cat']['onehot'].get_feature_names_out(input_features=categorical_cols)

# Combine categorical feature names with numerical feature names
all_feature_names = np.concatenate([numerical_cols, categorical_features])
```

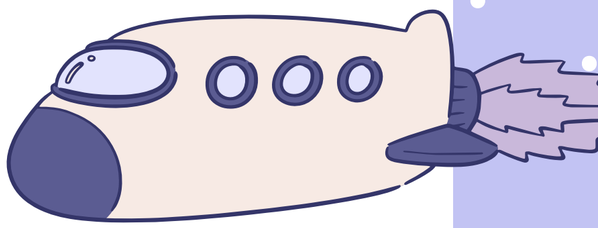
Random Forest

- Bagging (Bootstrap Aggregating)
- Random Feature Selection
- Ensemble Learning
- High Flexibility and Robustness
- Handle Missing Values
- Out-of-Bag (OOB) Error



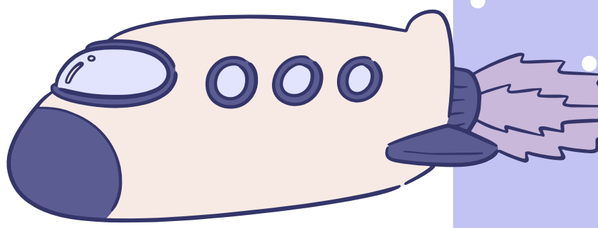


Random Forest



```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score

# Initializing the Random Forest Classifier
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Performing cross-validation to evaluate the model
cv_scores = cross_val_score(random_forest_model, X_train, y_train, cv=5, scoring='accuracy')

# Calculating the average cross-validation score
average_cv_score = cv_scores.mean()

average_cv_score
```

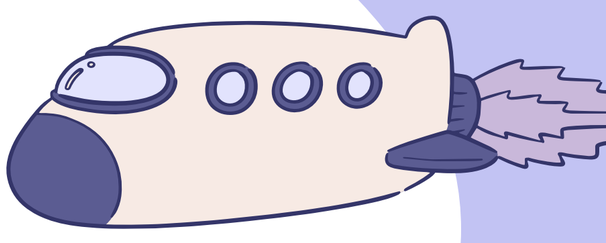
Random Forest

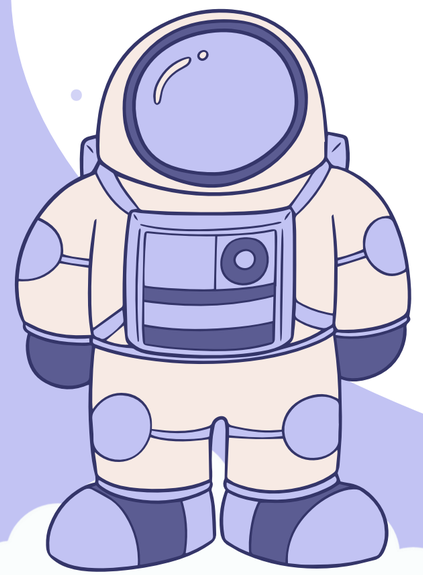


```
# Training the model on the entire training dataset
random_forest_model.fit(X_train, y_train)

# Making predictions on the test dataset
test_predictions = random_forest_model.predict(X_test)

# Preparing the submission file
submission_df = pd.DataFrame({'PassengerId': test_df['PassengerId'], 'Transported': test_predictions})
submission_df.head()

# Save the dataframe to a CSV file
submission_df.to_csv('submission.csv', index=False)

from google.colab import files
files.download('submission.csv')
```

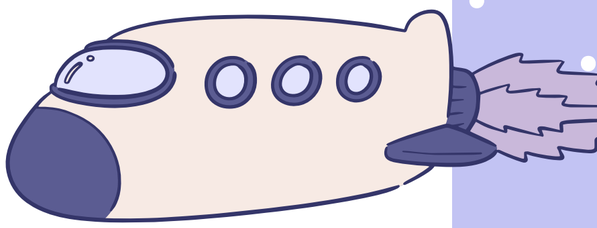
Decision Tree

- Random Forest
- Gradient Boosting Trees (GBT)
- AdaBoost (Adaptive Boosting)





Comparson



```
# Handling missing values
imputer = SimpleImputer(strategy='mean')
train_data_filled = pd.DataFrame(imputer.fit_transform(train_data.select_dtypes(include=['float64'])))
train_data_filled.columns = train_data.select_dtypes(include=['float64']).columns
train_data_filled.index = train_data.index

# Encoding categorical variables
train_data_encoded = pd.get_dummies(train_data.select_dtypes(include=['object', 'bool']))

# Combining numerical and categorical data
train_data_processed = pd.concat([train_data_filled, train_data_encoded], axis=1)

# Repeat the same for test_data
```

```
# Identify features and target variable
X = train_data_processed.drop(['Transported'], axis=1)  # Assuming 'Transported' is the target variable
y = train_data_processed['Transported']
```

```
# Splitting the dataset into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

Evaluation

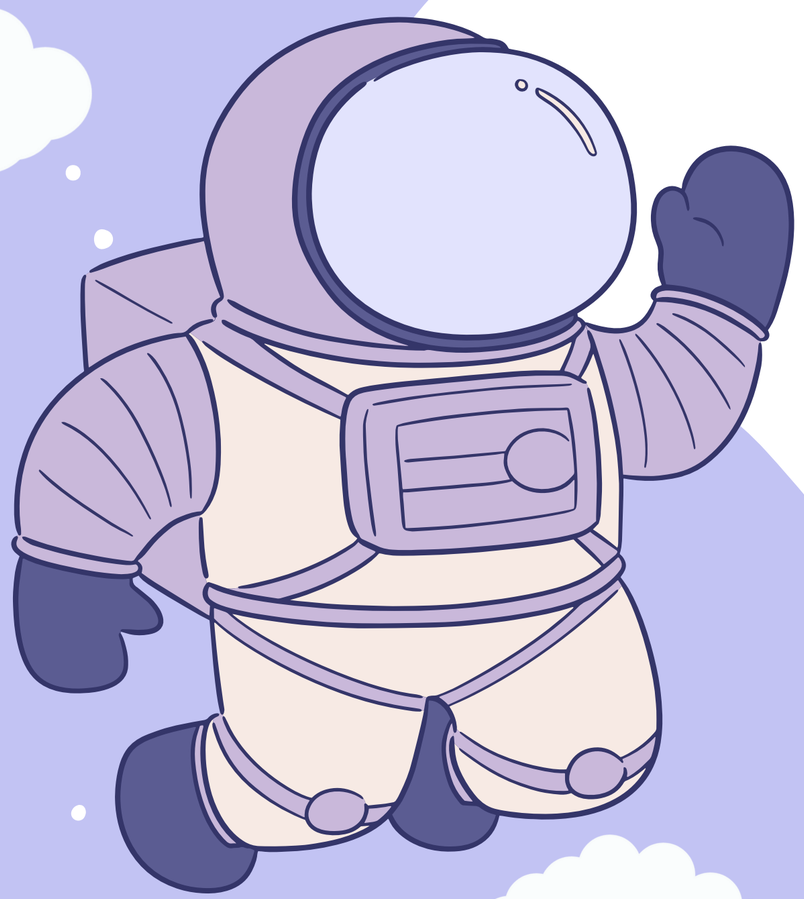
```python
from sklearn.tree import DecisionTreeClassifier

# Decision Tree
decision_tree_model = DecisionTreeClassifier(random_state=42)
decision_tree_model.fit(X_train, y_train)

# Random Forest
random_forest_model = RandomForestClassifier(random_state=42)
random_forest_model.fit(X_train, y_train)
```

```
▼ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Evaluation

```python
# Evaluating Decision Tree
y_pred_tree = decision_tree_model.predict(X_val)
print(f"Decision Tree Accuracy: {accuracy_score(y_val, y_pred_tree)}")

# Evaluating Random Forest
y_pred_forest = random_forest_model.predict(X_val)
print(f"Random Forest Accuracy: {accuracy_score(y_val, y_pred_forest)}")
```

```
Decision Tree Accuracy: 0.7745830937320299
Random Forest Accuracy: 0.7751581368602645
```



THANK YOU