GFG project

> **Name : Ayush gupta**

> **Registered Email : ayushgupta4044ayush@gmail.com**

Load the datasets

load the datasets into pandas DataFrames.

```
import pandas as pd

# Load the datasets
training_features = pd.read_csv('/mnt/data/training_set_featu
training_labels = pd.read_csv('/mnt/data/training_set_labels.
test_features = pd.read_csv('/mnt/data/test_set_features.csv'
submission_format = pd.read_csv('/mnt/data/submission_format.

# Display the first few rows of each DataFrame to understand
training_features.head(), training_labels.head(), test_featur
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncod
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.multioutput import MultiOutputClassifier
from sklearn.metrics import roc_auc_score

# Identify numerical and categorical columns
```

```
num_cols = training_features.select_dtypes(include=['int64',
cat_cols = training_features.select_dtypes(include=['object']

# Preprocessing for numerical data
num_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

# Preprocessing for categorical data
cat_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Bundle preprocessing for numerical and categorical data
preprocessor = ColumnTransformer(
    transformers=[
        ('num', num_transformer, num_cols),
        ('cat', cat_transformer, cat_cols)
    ])

# Define the model
model = MultiOutputClassifier(RandomForestClassifier(n_estima

# Create and evaluate the pipeline
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('model', model)])

# Split the data into training and validation sets
X_train, X_valid, y_train, y_valid = train_test_split(trainin

# Preprocess and fit the model
pipeline.fit(X_train, y_train)

# Make predictions
y_pred = pipeline.predict_proba(X_valid)
y_pred = pd.DataFrame({col: y_pred[i][:, 1] for i, col in enu
```

```
# Evaluate the model
roc_auc_score(y_valid, y_pred, average='macro')
```

```
# Make predictions on the test set
test_predictions = pipeline.predict_proba(test_features)
test_predictions = pd.DataFrame({
    'respondent_id': test_features['respondent_id'],
    'xyz_vaccine': test_predictions[0][:, 1],
    'seasonal_vaccine': test_predictions[1][:, 1]
})

# Save the predictions to a CSV file
test_predictions.to_csv('/mnt/data/submission.csv', index=Fal
```