ML_Pipeline\main.py

```python
# #imports
import pandas as pd
import numpy as np
import joblib



from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler , OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score,accuracy_score




# loading Data

data=pd.read_csv('C:\\Users\\morea\\Desktop\\ML\\ML_Pipeline\\CO2_emission.csv')
print(data.head())
data.info()
# create features and target variable
# Create features and target variable
x = data.drop(['CO2_Emissions'], axis=1)  # Correct column name
y = data['CO2_Emissions']  # Correct column name

# Split categorical and numerical
numerical_cols = ['Model_Year', 'Engine_Size', 'Cylinders',
                  'Fuel_Consumption_in_City(L/100 km)',
                  'Fuel_Consumption_in_City_Hwy(L/100 km)',
                  'Fuel_Consumption_comb(L/100km)', 'Smog_Level']
categorical_cols = ['Make', 'Model', 'Vehicle_Class', 'Transmission']

print(numerical_cols)
print(categorical_cols)
# Rest of your pipeline steps...



numerical_pipeline=Pipeline([
    ('imputer',SimpleImputer(strategy='mean')),
    ('scaler',StandardScaler())
])

categorical_pipeline=Pipeline([
    ('imputer',SimpleImputer(strategy='most_frequent')),
    ('encoder',OneHotEncoder(handle_unknown='ignore')) #convert all strings into binary
```

```
])

#join the pipelines together # combine the pipelines
preprocessor=ColumnTransformer([
    ('num',numerical_pipeline,numerical_cols),
    ('cat',categorical_pipeline,categorical_cols)
])
pipeline=Pipeline([
    ('preprocessor',preprocessor),
    ('model',RandomForestRegressor()),
])


# split into train and test
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)

# train and predict
pipeline.fit(x_train,y_train)

prediction=pipeline.predict(x_test)
# evalute accuract
mse=mean_squared_error(y_test,prediction)
rmse=np.sqrt(mse)
r2=r2_score(y_test,prediction)
print(mse,rmse,r2)

```