# ===============================================================

# Future job market predictor

# ===============================================================

# Description:

# This project uses technologies like

# HTML5+CSS3+javaScript

# ===============================================================

1. **data preprocessing.py:**

import pandas as pd

from sklearn.preprocessing import MinMaxScaler, OneHotEncoder

def load\_data(file\_path):

return pd.read\_csv(file\_path)

def preprocess\_data(df):

# Handle missing values

df = df.fillna(df.median())

# Encode categorical variables

categorical\_cols = ['Job\_Title', 'Sector']

df = pd.get\_dummies(df, columns=categorical\_cols, drop\_first=True)

# Scale numerical features

scaler = MinMaxScaler()

numeric\_cols = ['Employment\_Rate', 'Average\_Salary', 'Job\_Growth\_Rate']

df[numeric\_cols] = scaler.fit\_transform(df[numeric\_cols])

return df, scaler

1. **models.py:**

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Dropout

# Linear Regression

def build\_linear\_regression():

return LinearRegression()

# Random Forest Regressor

def build\_random\_forest(n\_estimators=200, max\_depth=12):

return RandomForestRegressor(n\_estimators=n\_estimators, max\_depth=max\_depth, random\_state=42)

# LSTM Model

def build\_lstm(input\_shape):

model = Sequential()

model.add(LSTM(128, activation='relu', input\_shape=input\_shape, return\_sequences=True))

model.add(LSTM(64, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

return model

1. **train.py**

from sklearn.model\_selection import train\_test\_split

from models import build\_linear\_regression, build\_random\_forest, build\_lstm

from data\_preprocessing import load\_data, preprocess\_data

import numpy as np

# Load and preprocess

df = load\_data('data/job\_data.csv')

data, scaler = preprocess\_data(df)

X = data.drop('Job\_Growth\_Rate', axis=1)

y = data['Job\_Growth\_Rate']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Linear Regression

lr\_model = build\_linear\_regression()

lr\_model.fit(X\_train, y\_train)

# Random Forest

rf\_model = build\_random\_forest()

rf\_model.fit(X\_train, y\_train)

# LSTM requires 3D input (samples, timesteps, features)

X\_train\_lstm = np.expand\_dims(X\_train.values, axis=1)

X\_test\_lstm = np.expand\_dims(X\_test.values, axis=1)

lstm\_model = build\_lstm(input\_shape=(X\_train\_lstm.shape[1], X\_train\_lstm.shape[2]))

lstm\_model.fit(X\_train\_lstm, y\_train, epochs=100, batch\_size=32, validation\_split=0.1)

1. **main.py:**

from train import lr\_model, rf\_model, lstm\_model, X\_test, y\_test

from evaluate import evaluate\_model, plot\_forecast

import numpy as np

# Linear Regression Evaluation

y\_pred\_lr = lr\_model.predict(X\_test)

evaluate\_model(y\_test, y\_pred\_lr, "Linear Regression")

plot\_forecast(y\_test, y\_pred\_lr, "Linear Regression")

# Random Forest Evaluation

y\_pred\_rf = rf\_model.predict(X\_test)

evaluate\_model(y\_test, y\_pred\_rf, "Random Forest")

plot\_forecast(y\_test, y\_pred\_rf, "Random Forest")

# LSTM Evaluation

X\_test\_lstm = np.expand\_dims(X\_test.values, axis=1)

y\_pred\_lstm = lstm\_model.predict(X\_test\_lstm)

evaluate\_model(y\_test, y\_pred\_lstm, "LSTM")

plot\_forecast(y\_test, y\_pred\_lstm, "LSTM")