Project Title: Big Sales Prediction using Random Forest Regressor

**1. Introduction**

* **Objective**: To predict sales figures using historical sales data and various features that could influence sales.
* **Importance**: Accurate sales predictions help businesses manage inventory, allocate resources efficiently, and develop better marketing strategies.

**2. Data Collection**

* **Source**: Acquire data from sources such as Kaggle, retail databases, or generated datasets.
* **Features**: Include relevant features like product ID, store ID, historical sales data, promotional events, pricing, seasonality, etc.
* **Tools**: Use Python libraries such as pandas for data manipulation.

**3. Data Preprocessing**

* **Handling Missing Values**: Fill or drop missing values using appropriate methods.
* **Feature Engineering**: Create new features that could enhance the model’s performance (e.g., moving averages, lag features).
* **Encoding Categorical Variables**: Use techniques like One-Hot Encoding for categorical features.
* **Scaling Features**: Normalize or standardize numerical features if necessary.

**4. Exploratory Data Analysis (EDA)**

* **Visualizations**: Use matplotlib and seaborn to create plots and understand data distributions.
* **Insights**: Draw insights from the data to guide the feature selection and engineering process.

**5. Model Building**

* **Splitting Data**: Split the data into training and testing sets.
* **Random Forest Regressor**: Implement the Random Forest Regressor using scikit-learn.
* **Hyperparameter Tuning**: Use techniques like GridSearchCV or RandomizedSearchCV to optimize model parameters.

**6. Model Evaluation**

* **Metrics**: Use evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared to assess model performance.
* **Validation**: Validate the model using cross-validation techniques.

**7. Model Interpretation**

* **Feature Importance**: Determine the importance of different features in the prediction process.
* **Partial Dependence Plots**: Use these plots to understand the relationship between features and the target variable.

**8. Deployment**

* **Saving the Model**: Save the trained model using libraries like joblib or pickle.
* **Prediction API**: Develop a simple API using Flask or FastAPI to serve predictions.
* **User Interface**: Optionally, create a front-end interface for users to input data and get predictions.

**9. Conclusion**

* **Summary**: Summarize the findings, model performance, and any challenges faced during the project.
* **Future Work**: Suggest possible improvements or future directions for the project.

**10. References**

* List any references, datasets, and libraries used in the project.

**Implementation Steps**

* Here’s a brief implementation of some key steps using Python:
* **Data Preprocessing and Model Building**

**import pandas as pd**

**import numpy as np**

**from sklearn.model\_selection import train\_test\_split, GridSearchCV**

**from sklearn.ensemble import RandomForestRegressor**

**from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score**

**import joblib**

**# Load data**

**data = pd.read\_csv('sales\_data.csv')**

**# Preprocessing (example)**

**data.fillna(0, inplace=True) # Handling missing values**

**# Feature Engineering (example)**

**data['year'] = pd.DatetimeIndex(data['date']).year**

**data['month'] = pd.DatetimeIndex(data['date']).month**

**# Encoding categorical variables (example)**

**data = pd.get\_dummies(data, columns=['store\_id', 'product\_id'])**

**# Splitting data**

**X = data.drop(['sales', 'date'], axis=1)**

**y = data['sales']**

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

**# Model building**

**rf = RandomForestRegressor(random\_state=42)**

**param\_grid = {**

**'n\_estimators': [100, 200, 300],**

**'max\_depth': [10, 20, 30],**

**'min\_samples\_split': [2, 5, 10]**

**}**

**grid\_search = GridSearchCV(estimator=rf, param\_grid=param\_grid, cv=3, n\_jobs=-1, verbose=2)**

**grid\_search.fit(X\_train, y\_train)**

**# Best model**

**best\_rf = grid\_search.best\_estimator\_**

**# Evaluation**

**y\_pred = best\_rf.predict(X\_test)**

**mae = mean\_absolute\_error(y\_test, y\_pred)**

**mse = mean\_squared\_error(y\_test, y\_pred)**

**r2 = r2\_score(y\_test, y\_pred)**

**print(f'MAE: {mae}, MSE: {mse}, R2: {r2}')**

**# Save the model**

**joblib.dump(best\_rf, 'sales\_prediction\_model.joblib')**

Deployment using Flask

from flask import Flask, request, jsonify

import joblib

import pandas as pd

app = Flask(\_\_name\_\_)

model = joblib.load('sales\_prediction\_model.joblib')

@app.route('/predict', methods=['POST'])

def predict():

data = request.json

df = pd.DataFrame(data)

prediction = model.predict(df)

return jsonify({'prediction': prediction.tolist()})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**1. Sample Data**

Here's a sample dataset in CSV format. Save it as sales\_data.csv.

**date,store\_id,product\_id,sales,price,promotion**

**2023-01-01,1,A,100,20,0**

**2023-01-01,1,B,150,15,1**

**2023-01-01,2,A,200,20,0**

**2023-01-01,2,B,250,15,1**

**2023-01-02,1,A,110,20,0**

**2023-01-02,1,B,140,15,1**

**2023-01-02,2,A,210,20,0**

**2023-01-02,2,B,260,15,1**

**2. Project Structure**

Create the following project structure:

**sales\_prediction/**

**│**

**├── data/**

**│ └── sales\_data.csv**

**│**

**├── model/**

**│ └── train\_model.py**

**│**

**├── app/**

**│ └── app.py**

**│**

**└── requirements.txt**

3. Data Preprocessing and Model Building (train\_model.py)

import pandas as pd

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

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

import joblib

# Load data

data = pd.read\_csv('data/sales\_data.csv')

# Preprocessing

data['date'] = pd.to\_datetime(data['date'])

data['year'] = data['date'].dt.year

data['month'] = data['date'].dt.month

data['day'] = data['date'].dt.day

# Encoding categorical variables

data = pd.get\_dummies(data, columns=['store\_id', 'product\_id'])

# Splitting data

X = data.drop(['sales', 'date'], axis=1)

y = data['sales']

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

# Model building

rf = RandomForestRegressor(random\_state=42)

param\_grid = {

'n\_estimators': [100, 200, 300],

'max\_depth': [10, 20, 30],

'min\_samples\_split': [2, 5, 10]

}

grid\_search = GridSearchCV(estimator=rf, param\_grid=param\_grid, cv=3, n\_jobs=-1, verbose=2)

grid\_search.fit(X\_train, y\_train)

# Best model

best\_rf = grid\_search.best\_estimator\_

# Evaluation

y\_pred = best\_rf.predict(X\_test)

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'MAE: {mae}, MSE: {mse}, R2: {r2}')

# Save the model

joblib.dump(best\_rf, 'model/sales\_prediction\_model.joblib')

4. Deployment with Flask (app.py)

from flask import Flask, request, jsonify

import joblib

import pandas as pd

app = Flask(\_\_name\_\_)

model = joblib.load('model/sales\_prediction\_model.joblib')

@app.route('/predict', methods=['POST'])

def predict():

data = request.json

df = pd.DataFrame(data)

prediction = model.predict(df)

return jsonify({'prediction': prediction.tolist()})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

5. Requirements (requirements.txt)

Flask==2.0.2

joblib==1.1.0

numpy==1.21.2

pandas==1.3.3

scikit-learn==0.24.2

**6. Running the Project**

1. **Install dependencies**: Run pip install -r requirements.txt to install the required packages.
2. **Train the model**: Execute python model/train\_model.py to preprocess the data, train the model, and save it.
3. **Run the Flask app**: Start the server by running python app/app.py.

**7. Testing the API**

Use curl or Postman to send a POST request to the /predict endpoint.

Example:

curl -X POST -H "Content-Type: application/json" -d '[

{

"price": 20,

"promotion": 0,

"year": 2023,

"month": 1,

"day": 1,

"store\_id\_1": 1,

"store\_id\_2": 0,

"product\_id\_A": 1,

"product\_id\_B": 0

},

{

"price": 15,

"promotion": 1,

"year": 2023,

"month": 1,

"day": 2,

"store\_id\_1": 0,

"store\_id\_2": 1,

"product\_id\_A": 0,

"product\_id\_B": 1

}

]' <http://127.0.0.1:5000/predict>

You should receive a response with predicted sales:

{

"prediction": [110.0, 260.0]

}

This completes the project setup for big sales prediction using a Random Forest Regressor. Adjust the dataset and feature engineering according to your specific use case for better results.