Install necessary libraries using pip

```
pip install numpy
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.23.5)
```

```
pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
```

```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.14.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes==0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/pytho
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.34.
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.59.2)
Requirement already satisfied: tensorboard<2.15,>=2.14 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.1)
Requirement already satisfied: tensorflow-estimator<2.15,>=2.14.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.0)
Requirement already satisfied: keras<2.15,>=2.14.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.41
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14->tensorfl
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14-
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14->tensorflow) (3
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14->tensorflow
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14->tensorflow) (3
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboa
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboar
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.15,>
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<1.1,>=0.5
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboa
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.15,>=2.
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.1
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.15,>=2
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-o
```

Import required libraries

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
```

Read the dataset from a CSV file and display the first few rows

```
df = pd.read_csv('car_purchasing.csv',encoding='ISO-8859-1')
```

```
df.head()
```

	customer name	customer e-mail	country	gender	age
0	Martina Avila	cubilia.Curae.Phasellus@quisaccumsanconvallis.edu	Bulgaria	0	41.851720
1	Harlan Barnes	eu.dolor@diam.co.uk	Belize	0	40.870623
2	Naomi Rodriquez	vulputate.mauris.sagittis@ametconsectetueradip...	Algeria	1	43.152897
3	Jade Cunningham	malesuada@dignissim.com	Cook Islands	1	58.271369
4	Cedric Leach	felis.ullamcorper.viverra@egetmollislectus.net	Brazil	1	57.313749

Define a list of columns to drop from the DataFrame

```
columns_to_drop = ['customer name', 'customer e-mail', 'country']
new_df = df.drop(columns=columns_to_drop)
```

Show ChatGPT

```
new_df.head()
```

	gender	age	annual Salary	credit card debt	net worth	car purchase amount
0	0	41.851720	62812.09301	11609.380910	238961.2505	35321.45877
1	0	40.870623	66646.89292	9572.957136	530973.9078	45115.52566
2	1	43.152897	53798.55112	11160.355060	638467.1773	42925.70921
3	1	58.271369	79370.03798	14426.164850	548599.0524	67422.36313
4	1	57.313749	59729.15130	5358.712177	560304.0671	55915.46248

Apply Min-Max scaling to the 'age' column in the DataFrame

```
#standard_scaler = StandardScaler()
#new_df['age_standard_scaled'] = standard_scaler.fit_transform(new_df[['age']])
min_max_scaler = MinMaxScaler()
new_df['age'] = min_max_scaler.fit_transform(new_df[['age']])
```

Show ChatGPT

```
new_df.head()
```

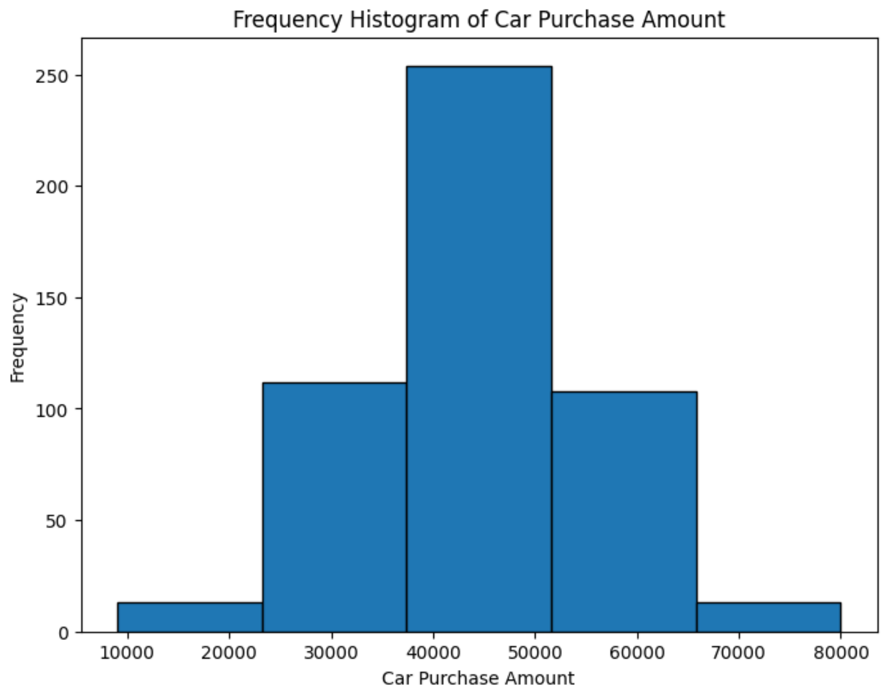
	gender	age	annual Salary	credit card debt	net worth	car purchase amount	age_standard_s
0	0	41.851720	62812.09301	11609.380910	238961.2505	35321.45877	-0.5

Import Matplotlib for data visualization

```
import matplotlib.pyplot as plt
import pandas as pd

# Plot the frequency histogram
plt.figure(figsize=(8, 6))
plt.hist(df['car purchase amount'], bins=5, edgecolor='k')
plt.xlabel('Car Purchase Amount')
plt.ylabel('Frequency')
plt.title('Frequency Histogram of Car Purchase Amount')
plt.show()
```

Show ChatGPT



Create a frequency histogram for the 'car purchase amount' column

Apply Standard Scaling to specific columns in the DataFrame

```
salary_scaler = StandardScaler()
debt_scaler = StandardScaler()
net_worth_scaler = StandardScaler()
new_df['annual Salary'] = salary_scaler.fit_transform(new_df[['annual Salary']])
new_df['credit card debt'] = debt_scaler.fit_transform(new_df[['credit card debt']])
new_df['net worth'] = net_worth_scaler.fit_transform(new_df[['net worth']])


```

Show ChatGPT

```
new_df.head()
```

	gender	age	annual Salary	credit card debt	net worth	car purchase amount	age_standard_scaled	age_min_max_scaled
0	0	0.437034	0.058576	0.574271	-1.110469	35321.45877	-0.550749	0.437034
1	0	0.417412	0.386570	-0.009951	0.573929	45115.52566	-0.673834	0.417412
2	1	0.463058	-0.712361	0.445452	1.193976	42925.70921	-0.387508	0.463058
3	1	0.765427	1.474794	1.382369	0.675595	67422.36313	1.509206	0.765427
4	1	0.746275	-0.205111	-1.218962	0.743113	55915.46248	1.389066	0.746275

+ Code + Text

```
new_df = new_df.iloc[:, :-2]

```

Show ChatGPT

```
new_df.head()
```

	gender	age	annual Salary	credit card debt	net worth	car purchase amount
0	0	0.437034	0.058576	0.574271	-1.110469	35321.45877
1	0	0.417412	0.386570	-0.009951	0.573929	45115.52566
2	1	0.463058	-0.712361	0.445452	1.193976	42925.70921
3	1	0.765427	1.474794	1.382369	0.675595	67422.36313
4	1	0.746275	-0.205111	-1.218962	0.743113	55915.46248

Create a Random Forest Regressor model and fit it to the data

Calculate and print feature importances in descending order

```
X = new_df.drop('car purchase amount', axis=1)
y = new_df['car purchase amount']
model = RandomForestRegressor()
model.fit(X, y)
feature_importances = model.feature_importances_
sorted_indices = feature_importances.argsort()[::-1]
top_features = X.columns[sorted_indices[:3]]
print(top_features)

```

```
Index(['age', 'annual Salary', 'net worth'], dtype='object')
```

Split the data into training and testing sets

```
from sklearn.model_selection import train_test_split
X = new_df[['age', 'annual Salary', 'net worth']]
y = new_df['car purchase amount']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

Show ChatGPT

Create a Sequential model for neural network

```
model =  Sequential()
model.add(Dense(2 , input_dim=3 , activation= 'relu' ))
model.add(Dense(1, activation = 'linear'))
```

Compile the neural network model

```
model.compile(
    loss = 'binary_crossentropy',
    optimizer = 'adam',
    metrics = ['accuracy']
)
```

Fit the model using the original data

```
orginal_X = df[['age', 'annual Salary', 'net worth']]
orginal_y = df['car purchase amount']
model.fit(
    orginal_X , orginal_y ,
    epochs = 50 ,
    batch_size = 3 ,
    verbose = 1
)

```

```
167/167 [==============================] - 0s 2ms/step - loss: -674150.4375 - accuracy: 0.0000e+00
Epoch 23/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.5625 - accuracy: 0.0000e+00
Epoch 24/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.5625 - accuracy: 0.0000e+00
Epoch 25/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.6250 - accuracy: 0.0000e+00
Epoch 26/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.2500 - accuracy: 0.0000e+00
Epoch 27/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.5625 - accuracy: 0.0000e+00
Epoch 28/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.3125 - accuracy: 0.0000e+00
Epoch 29/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.3750 - accuracy: 0.0000e+00
Epoch 30/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.3750 - accuracy: 0.0000e+00
Epoch 31/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.4375 - accuracy: 0.0000e+00
Epoch 32/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.5000 - accuracy: 0.0000e+00
Epoch 33/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.5000 - accuracy: 0.0000e+00
Epoch 34/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.6250 - accuracy: 0.0000e+00
Epoch 35/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.3125 - accuracy: 0.0000e+00
Epoch 36/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.6250 - accuracy: 0.0000e+00
Epoch 37/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.3750 - accuracy: 0.0000e+00
Epoch 38/50
167/167 [==============================] - 1s 3ms/step - loss: -674150.3750 - accuracy: 0.0000e+00
Epoch 39/50
167/167 [==============================] - 1s 3ms/step - loss: -674150.8125 - accuracy: 0.0000e+00
Epoch 40/50
167/167 [==============================] - 1s 3ms/step - loss: -674150.5625 - accuracy: 0.0000e+00
Epoch 41/50
167/167 [==============================] - 1s 3ms/step - loss: -674150.7500 - accuracy: 0.0000e+00
Epoch 42/50
167/167 [==============================] - 1s 3ms/step - loss: -674150.5000 - accuracy: 0.0000e+00
Epoch 43/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.5625 - accuracy: 0.0000e+00
Epoch 44/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.3750 - accuracy: 0.0000e+00
Epoch 45/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.8750 - accuracy: 0.0000e+00
Epoch 46/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.8125 - accuracy: 0.0000e+00
Epoch 47/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.5000 - accuracy: 0.0000e+00
Epoch 48/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.5625 - accuracy: 0.0000e+00
Epoch 49/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.3125 - accuracy: 0.0000e+00
Epoch 50/50
167/167 [==============================] - 0s 2ms/step - loss: -674150.6250 - accuracy: 0.0000e+00
<keras.src.callbacks.History at 0x7e0bdb354370>
```

Fit the model using the training data

```
model.fit(
    X_train , y_train ,
    epochs = 50 ,
    batch_size = 3 ,
    verbose = 1
)
```

```
Epoch 1/50
134/134 [==============================] - 2s 7ms/step - loss: -670128.9375 - accuracy: 0.0000e+00
Epoch 2/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.8125 - accuracy: 0.0000e+00
Epoch 3/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.8750 - accuracy: 0.0000e+00
Epoch 4/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.6875 - accuracy: 0.0000e+00
Epoch 5/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.7500 - accuracy: 0.0000e+00
Epoch 6/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.6875 - accuracy: 0.0000e+00
Epoch 7/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.8750 - accuracy: 0.0000e+00
Epoch 8/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.7500 - accuracy: 0.0000e+00
Epoch 9/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.8750 - accuracy: 0.0000e+00
Epoch 10/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.7500 - accuracy: 0.0000e+00
Epoch 11/50
134/134 [==============================] - 0s 2ms/step - loss: -670129.0000 - accuracy: 0.0000e+00
Epoch 12/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.9375 - accuracy: 0.0000e+00
Epoch 13/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.8750 - accuracy: 0.0000e+00
Epoch 14/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.9375 - accuracy: 0.0000e+00
Epoch 15/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.8750 - accuracy: 0.0000e+00
Epoch 16/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.9375 - accuracy: 0.0000e+00
Epoch 17/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.7500 - accuracy: 0.0000e+00
Epoch 18/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.7500 - accuracy: 0.0000e+00
Epoch 19/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.9375 - accuracy: 0.0000e+00
Epoch 20/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.8125 - accuracy: 0.0000e+00
Epoch 21/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.8750 - accuracy: 0.0000e+00
Epoch 22/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.7500 - accuracy: 0.0000e+00
Epoch 23/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.8125 - accuracy: 0.0000e+00
Epoch 24/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.8750 - accuracy: 0.0000e+00
Epoch 25/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.7500 - accuracy: 0.0000e+00
Epoch 26/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.7500 - accuracy: 0.0000e+00
Epoch 27/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.8750 - accuracy: 0.0000e+00
Epoch 28/50
134/134 [==============================] - 0s 2ms/step - loss: -670129.0000 - accuracy: 0.0000e+00
Epoch 29/50
134/134 [==============================] - 0s 2ms/step - loss: -670128.9375 - accuracy: 0.0000e+00
```

Evaluate the model on the testing data

```
model.evaluate(X_test,y_test,verbose=1)
```

```
4/4 [==============================] - 0s 4ms/step - loss: -690237.1250 - accuracy: 0.0000e+00
[-690237.125, 0.0]
```

Create a Random Forest Regressor model with 100 estimators and fit it to the training data

```
random_forest = RandomForestRegressor(n_estimators=100)
random_forest.fit(X_train,y_train)


```

```
    0.96004468314316
```

Make predictions using the random forest model and calculate the R-squared score

```
random_f_pred = random_forest.predict(X_test)
r2_score(y_test,random_f_pred
```