```
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


# Load data
df = pd.read_csv('CompleteResponses.csv', delimiter=',')
df.shape
df.head(10)
```

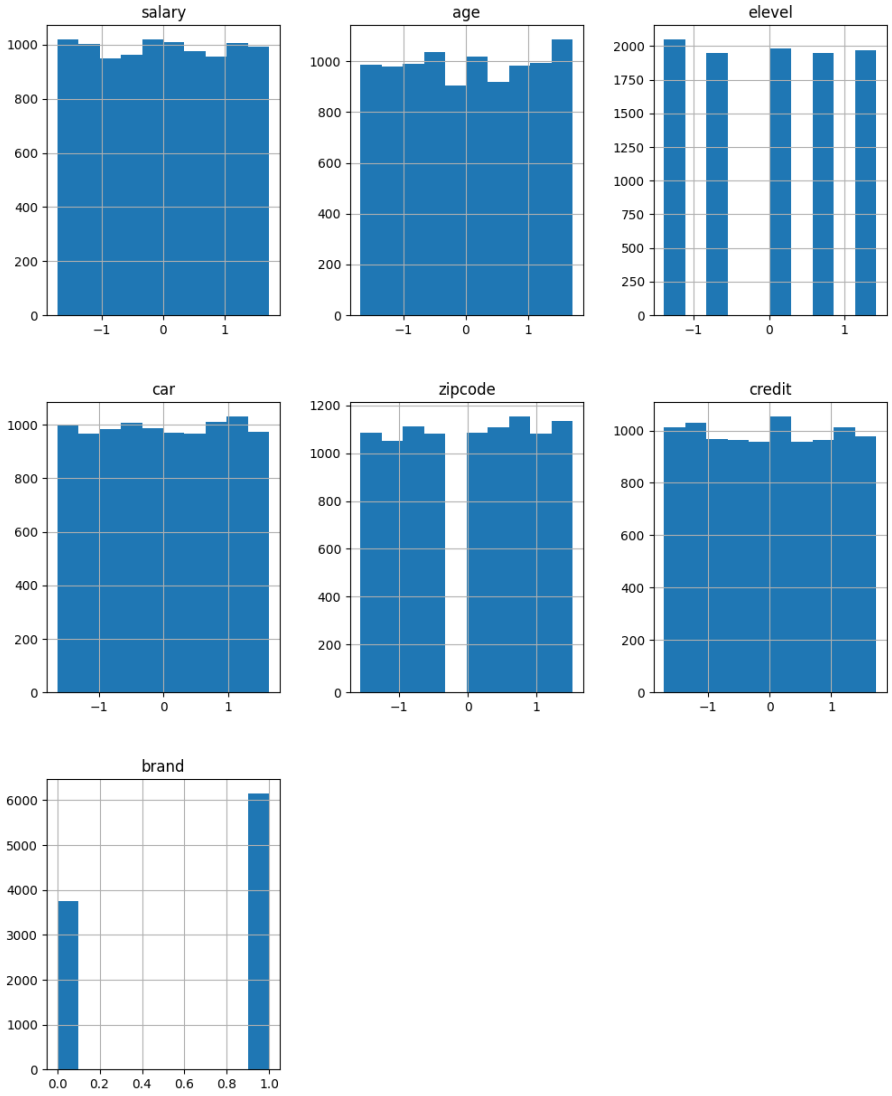
	salary	age	elevel	car	zipcode	credit	brand
0	119806.54480	45	0	14	4	442037.71130	0
1	106880.47840	63	1	11	6	45007.17883	1
2	78020.75094	23	0	15	2	48795.32279	0
3	63689.93635	51	3	6	5	40888.87736	1
4	50873.61880	20	3	14	4	352951.49770	0
5	130812.74280	56	3	14	3	135943.02200	1
6	136459.33920	24	4	8	5	80500.56351	1
7	103866.89960	62	3	3	0	359803.89350	1
8	72298.80402	29	4	17	0	276298.69520	0
9	37803.33285	41	1	5	4	493219.26860	1

```
# Normalize data

# Remove the target column from the DataFrame
target_col = 'brand'
X = df.drop(columns=[target_col])

# Compute the mean and standard deviation of each column
means = X.mean()
stds = X.std()

# Subtract the mean and divide by the standard deviation for each value in each column
for col in X.columns:
    X[col] = (X[col] - means[col]) / stds[col]

# Add the target column back to the DataFrame
df_norm = pd.concat([X, df[target_col]], axis=1)



df_norm.head()
```

	salary	age	elevel	car	zipcode	credit	brand
0	0.926373	-0.271565	-1.395874	0.602636	-0.015808	1.328143	0
1	0.583618	0.751355	-0.691927	0.082835	0.758768	-1.406009	1
2	-0.181641	-1.521800	-1.395874	0.775903	-0.790384	-1.379922	0
3	-0.561644	0.069408	0.715966	-0.783501	0.371480	-1.434370	1
4	-0.901489	-1.692287	0.715966	0.602636	-0.015808	0.714651	0

```
# Plot distributions
df_norm.hist(figsize=(12,15))
df_norm.head()
```

	salary	age	elevel	car	zipcode	credit	brand
0	0.926373	-0.271565	-1.395874	0.602636	-0.015808	1.328143	0
1	0.583618	0.751355	-0.691927	0.082835	0.758768	-1.406009	1
2	-0.181641	-1.521800	-1.395874	0.775903	-0.790384	-1.379922	0
3	-0.561644	0.069408	0.715966	-0.783501	0.371480	-1.434370	1
4	-0.901489	-1.692287	0.715966	0.602636	-0.015808	0.714651	0




```
# Split into features and labels
X = df_norm.iloc[:, :-1].values
y = df_norm.iloc[:, -1].values


# Build multilayer model to overfit data
model = Sequential()

model.add(Dense(1, activation='sigmoid', input_dim=df_norm.shape[1]-1))

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

```
model.fit(X, y, epochs=200, verbose=1)
```

```
310/310 [==============================] - 1s 2ms/step - loss: 0.1664 - accuracy: 0.9253
Epoch 47/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1673 - accuracy: 0.9263
Epoch 48/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1653 - accuracy: 0.9247
Epoch 49/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1667 - accuracy: 0.9268
Epoch 50/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1655 - accuracy: 0.9258
Epoch 51/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1662 - accuracy: 0.9250
Epoch 52/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1654 - accuracy: 0.9261
Epoch 53/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1655 - accuracy: 0.9256
Epoch 54/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1657 - accuracy: 0.9235
Epoch 55/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1638 - accuracy: 0.9258
Epoch 56/200
310/310 [==============================] - 1s 3ms/step - loss: 0.1652 - accuracy: 0.9253
Epoch 57/200
310/310 [==============================] - 1s 3ms/step - loss: 0.1649 - accuracy: 0.9250
Epoch 58/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1645 - accuracy: 0.9266
Epoch 59/200
310/310 [==============================] - 1s 3ms/step - loss: 0.1643 - accuracy: 0.9253
Epoch 60/200
310/310 [==============================] - 1s 3ms/step - loss: 0.1652 - accuracy: 0.9264
Epoch 61/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1640 - accuracy: 0.9244
Epoch 62/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1649 - accuracy: 0.9262
Epoch 63/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1635 - accuracy: 0.9252
Epoch 64/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1634 - accuracy: 0.9262
Epoch 65/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1642 - accuracy: 0.9250
Epoch 66/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1626 - accuracy: 0.9269
Epoch 67/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1638 - accuracy: 0.9241
Epoch 68/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1626 - accuracy: 0.9272
Epoch 69/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1629 - accuracy: 0.9263
Epoch 70/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1631 - accuracy: 0.9266
Epoch 71/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1642 - accuracy: 0.9272
Epoch 72/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1624 - accuracy: 0.9254
Epoch 73/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1629 - accuracy: 0.9264
Epoch 74/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1635 - accuracy: 0.9264
Epoch 75/200
310/310 [==============================] - 1s 2ms/step - loss: 0.1635 - accuracy: 0.9260
```

```
# Build multilayer model to overfit data
model = Sequential()

model.add(Dense(16, activation='relu', input_dim=df_norm.shape[1]-1))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

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


model.fit(X, y, epochs=100, verbose=1)
```

```
Epoch 1/100
310/310 [==============================] - 1s 2ms/step - loss: 0.6881 - accuracy: 0.5814
Epoch 2/100
310/310 [==============================] - 1s 2ms/step - loss: 0.5845 - accuracy: 0.6692
Epoch 3/100
310/310 [==============================] - 1s 2ms/step - loss: 0.5438 - accuracy: 0.7343
Epoch 4/100
310/310 [==============================] - 1s 2ms/step - loss: 0.4986 - accuracy: 0.7942
```

```
Epoch 5/100
310/310 [==============================] - 1s 2ms/step - loss: 0.4351 - accuracy: 0.8376
Epoch 6/100
310/310 [==============================] - 1s 2ms/step - loss: 0.3729 - accuracy: 0.8719
Epoch 7/100
310/310 [==============================] - 1s 2ms/step - loss: 0.3307 - accuracy: 0.8911
Epoch 8/100
310/310 [==============================] - 1s 2ms/step - loss: 0.3046 - accuracy: 0.8923
Epoch 9/100
310/310 [==============================] - 1s 3ms/step - loss: 0.2873 - accuracy: 0.8954
Epoch 10/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2746 - accuracy: 0.8969
Epoch 11/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2652 - accuracy: 0.8996
Epoch 12/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2577 - accuracy: 0.8990
Epoch 13/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2512 - accuracy: 0.9016
Epoch 14/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2461 - accuracy: 0.9020
Epoch 15/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2415 - accuracy: 0.9021
Epoch 16/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2379 - accuracy: 0.9014
Epoch 17/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2345 - accuracy: 0.9032
Epoch 18/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2312 - accuracy: 0.9038
Epoch 19/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2285 - accuracy: 0.9041
Epoch 20/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2272 - accuracy: 0.9052
Epoch 21/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2236 - accuracy: 0.9053
Epoch 22/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2221 - accuracy: 0.9051
Epoch 23/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2200 - accuracy: 0.9057
Epoch 24/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2176 - accuracy: 0.9060
Epoch 25/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2180 - accuracy: 0.9065
Epoch 26/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2150 - accuracy: 0.9075
Epoch 27/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2137 - accuracy: 0.9059
Epoch 28/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2120 - accuracy: 0.9083
Epoch 29/100
310/310 [==============================] - 1s 2ms/step - loss: 0.2114 - accuracy: 0.9081
```