```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from keras.models import Sequential
from keras.layers import Dense, Dropout, BatchNormalization
from keras.optimizers import Adam
from sklearn.metrics import mean_squared_error
from keras.callbacks import EarlyStopping


train_data=pd.read_csv('/content/train_data.csv')
test_data=pd.read_csv('/content/test_data.csv')


train_data.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614

```
# PREPROCESSING
train_data.isnull().sum()
```

```
Item_Identifier               0
Item_Weight                1463
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                2410
Outlet_Location_Type          0
Outlet_Type                   0
Item_Outlet_Sales             0
dtype: int64
```

```
test_data.isnull().sum()
```

```
Item_Identifier               0
Item_Weight                 976
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                1606
Outlet_Location_Type          0
Outlet_Type                   0
dtype: int64
```

```
train_data['Item_Weight'].fillna(train_data['Item_Weight'].mean(), inplace=True)
train_data['Outlet_Size'].fillna(train_data['Outlet_Size'].mode()[0], inplace=True)
test_data['Item_Weight'].fillna(test_data['Item_Weight'].mean(), inplace=True)
test_data['Outlet_Size'].fillna(test_data['Outlet_Size'].mode()[0], inplace=True)


target_column = 'Item_Outlet_Sales'
train_data[target_column] = (train_data[target_column] - train_data[target_column].mean()) / train_data[target_column].std()
```

```
label_encoders = {}
categorical_columns = ['Item_Fat_Content', 'Item_Type', 'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type']
for col in categorical_columns:
    le = LabelEncoder()
    train_data[col] = le.fit_transform(train_data[col])
    test_data[col] = le.transform(test_data[col])
    label_encoders[col] = le


X = train_data.drop('Item_Outlet_Sales', axis=1)
y = train_data['Item_Outlet_Sales']


X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)


# Remove identifier columns before scaling
X_train_features = X_train.drop(['Item_Identifier', 'Outlet_Identifier'], axis=1)
X_val_features = X_val.drop(['Item_Identifier', 'Outlet_Identifier'], axis=1)


# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_features)
X_val_scaled = scaler.transform(X_val_features)


# Build the neural network model
model = Sequential()
model.add(Dense(256, activation='relu', input_dim=X_train_scaled.shape[1]))
model.add(BatchNormalization())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(64, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(1)) # Regression task, so no activation
# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error')
# Implement early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True)
# Train the model
s=model.fit(X_train_scaled, y_train, epochs=150, batch_size=64, validation_data=(X_val_scaled, y_val), callbacks=[early_stopping])
# Make predictions on validation data
y_pred = model.predict(X_val_scaled)
# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_val, y_pred))
print("RMSE:", rmse)
```

```
Epoch 1/150
107/107 [==============================] - 7s 20ms/step - loss: 0.7917 - val_loss: 0.6673
Epoch 2/150
107/107 [==============================] - 1s 11ms/step - loss: 0.5364 - val_loss: 0.5591
Epoch 3/150
107/107 [==============================] - 2s 15ms/step - loss: 0.4867 - val_loss: 0.4396
Epoch 4/150
107/107 [==============================] - 1s 8ms/step - loss: 0.4802 - val_loss: 0.3930
Epoch 5/150
107/107 [==============================] - 1s 8ms/step - loss: 0.4708 - val_loss: 0.3749
Epoch 6/150
107/107 [==============================] - 1s 6ms/step - loss: 0.4543 - val_loss: 0.3807
Epoch 7/150
107/107 [==============================] - 0s 5ms/step - loss: 0.4528 - val_loss: 0.3868
Epoch 8/150
107/107 [==============================] - 1s 5ms/step - loss: 0.4503 - val_loss: 0.3758
Epoch 9/150
107/107 [==============================] - 0s 5ms/step - loss: 0.4386 - val_loss: 0.3721
Epoch 10/150
107/107 [==============================] - 1s 5ms/step - loss: 0.4393 - val_loss: 0.3697
Epoch 11/150
107/107 [==============================] - 1s 5ms/step - loss: 0.4368 - val_loss: 0.3767
Epoch 12/150
107/107 [==============================] - 1s 5ms/step - loss: 0.4371 - val_loss: 0.3704
Epoch 13/150
107/107 [==============================] - 1s 5ms/step - loss: 0.4316 - val_loss: 0.3864
Epoch 14/150
107/107 [==============================] - 1s 5ms/step - loss: 0.4277 - val_loss: 0.3902
Epoch 15/150
107/107 [==============================] - 1s 5ms/step - loss: 0.4257 - val_loss: 0.3671
Epoch 16/150
107/107 [==============================] - 1s 5ms/step - loss: 0.4234 - val_loss: 0.3928
Epoch 17/150
```

```
107/107 [==============================] - 1s 10ms/step - loss: 0.4210 - val_loss: 0.3847
Epoch 18/150
107/107 [==============================] - 1s 9ms/step - loss: 0.4227 - val_loss: 0.3636
Epoch 19/150
107/107 [==============================] - 1s 8ms/step - loss: 0.4231 - val_loss: 0.3661
Epoch 20/150
107/107 [==============================] - 1s 9ms/step - loss: 0.4220 - val_loss: 0.3690
Epoch 21/150
107/107 [==============================] - 1s 8ms/step - loss: 0.4177 - val_loss: 0.3693
Epoch 22/150
107/107 [==============================] - 1s 7ms/step - loss: 0.4247 - val_loss: 0.3672
Epoch 23/150
107/107 [==============================] - 1s 7ms/step - loss: 0.4200 - val_loss: 0.3743
Epoch 24/150
107/107 [==============================] - 1s 7ms/step - loss: 0.4238 - val_loss: 0.3692
Epoch 25/150
107/107 [==============================] - 1s 6ms/step - loss: 0.4103 - val_loss: 0.3624
Epoch 26/150
107/107 [==============================] - 0s 5ms/step - loss: 0.4097 - val_loss: 0.3753
Epoch 27/150
107/107 [==============================] - 1s 5ms/step - loss: 0.4155 - val_loss: 0.3687
Epoch 28/150
107/107 [==============================] - 1s 6ms/step - loss: 0.4135 - val_loss: 0.3720
Epoch 29/150
```

```
print("RMSE:", rmse)
```

```
RMSE: 0.602028250096658
```

```
test_data = test_data.drop(['Item_Identifier', 'Outlet_Identifier'], axis=1)
```

```
from keras.models import load_model
model_filename = 'hello.h5'
model.save(model_filename)
print(f"Model saved as '{model_filename}'")
```

```
Model saved as 'hello.h5'
```

```
import os
print("Current working directory:", os.getcwd())
```

```
Current working directory: /content
```

```
from keras.models import load_model

loaded_model = load_model('/content/hello.h5')
```

```
from keras.models import load_model
loaded_model = load_model('/content/hello.h5')
# Make predictions using the loaded model
predictions = loaded_model.predict(test_data)
```

```
178/178 [==============================] - 0s 2ms/step
```

```
test_data['predicted_sales'] = predictions
test_data=pd.DataFrame(test_data)
```

```
test_data.head()
```

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Ou
0	20.750000	1	0.007565	13	107.8622	1999	1	0	
1	8.300000	4	0.038428	4	87.3198	2007	1	1	
2	14.600000	1	0.099575	11	241.7538	1998	1	2	
3	7.315000	1	0.015388	13	155.0340	2007	1	1	
4	12.695633	2	0.118599	4	234.2300	1985	1	2	

```
test_data
```

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type
0	20.750000	1	0.007565	13	107.8622	1999	1	0
1	8.300000	4	0.038428	4	87.3198	2007	1	1
2	14.600000	1	0.099575	11	241.7538	1998	1	2
3	7.315000	1	0.015388	13	155.0340	2007	1	1
4	12.695633	2	0.118599	4	234.2300	1985	1	2
...	...	...	...	...	...	...	...	...
5676	10.500000	2	0.013496	13	141.3154	1997	2	0
5677	7.600000	2	0.142991	15	169.1448	2009	1	2
5678	10.000000	1	0.073529	8	118.7440	2002	1	1
5679	15.300000	2	0.000000	3	214.6218	2007	1	1
5680	9.500000	2	0.104720	3	79.7960	2002	1	1

✓ 0s completed at 10:52 PM