```
from sklearn.metrics import r2_score


# Import required libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import RootMeanSquaredError


# load the dataset
df = pd.read_csv('/content/Train (1).csv')


# Drop the unnecessary columns
df = df.drop(['Item_Identifier', 'Outlet_Identifier'], axis=1)


# Replace missing values in Item_Weight with the mean value
df['Item_Weight'].fillna(df['Item_Weight'].mean(), inplace=True)


# Replace missing values in Outlet_Size with 'Unknown'
df['Outlet_Size'].fillna('Unknown', inplace=True)


# Convert categorical variables into numerical variables using one-hot encoding
df = pd.get_dummies(df)


# split the dataset into training and testing sets
X = df.drop('Item_Outlet_Sales', axis=1)
y = df['Item_Outlet_Sales']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Standardize the input features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
from sklearn.metrics import mean_squared_error
import math
```

Linear Regression Model

accuracy is 0.579

```
from sklearn.linear_model import LinearRegression
```

```
# create a linear regression model
model = LinearRegression()
```

```
# train the model
model.fit(X_train, y_train)
```

▾ LinearRegression

LinearRegression()

```
# make predictions on the testing set
y_pred = model.predict(X_test)
```

```
# calculate the accuracy of the model
accuracy = r2_score(y_test, y_pred)
```

```
print("Accuracy : ",accuracy);
```

```
Accuracy :  0.5792143884450605
```

```
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print('RMSE: ', rmse)
```

```
RMSE:  1069.4310153859096
```

XGBoost Regression

Accuracy is 0.539

```
# import xgboost as xgb


# model = xgb.XGBRegressor(n_estimators=500, learning_rate=0.1, max_depth=5, objective='reg:squarederror', random_state=42)


# model.fit(X_train, y_train)


# y_pred = model.predict(X_test)


# rmse = np.sqrt(mean_squared_error(y_test, y_pred))
# print('RMSE: ', rmse)


# accuracy = r2_score(y_test, y_pred)
# print("Accuracy is:", accuracy)
```

Artificial Neural Network

Accuracy is 0.507

```
# # Define the ANN model
# model = Sequential()
```

```
# model.add(Dense(units=64, activation='relu', input_shape=(X_train.shape[1],)))
# model.add(BatchNormalization())
# model.add(Dense(units=32, activation='relu'))
# model.add(BatchNormalization())
# model.add(Dense(units=16, activation='relu'))
# model.add(BatchNormalization())
# model.add(Dense(units=1, activation='linear'))


# from sklearn import metrics
# # Compile the model
# model.compile(optimizer=Adam(learning_rate=0.01), loss='mse', metrics=["accuracy"] )


# # Train the model
# history = model.fit(X_train, y_train, batch_size=64, epochs=100, validation_data=(X_test, y_test))


# from sklearn.metrics import mean_squared_error
# import math


# # Evaluate the model
# y_pred = model.predict(X_test)
# rmse = np.sqrt(mean_squared_error(y_test, y_pred))
# print('RMSE: ', rmse)


# # Evaluate the model on the testing set
# y_pred = model.predict(X_test)
# accuracy = r2_score(y_test, y_pred)
# print("Accuracy is:", accuracy)


# # Save the predicted values to a CSV file
# predicted_df = pd.DataFrame({'Item_Outlet_Sales_Predicted': y_pred.reshape(-1)})
# predicted_df.to_csv('big_mart_sales_predicted.csv', index=False)
```

10s completed at 9:07 PM