Mall Customer Behavior Analysis – OGT Project

```
#Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
#Load Dataset
df = pd.read_csv("Mall_Customers.csv")
```

Exploratory Data Analysis (EDA)

```
#printing first five rows
df.head()
```

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Next steps: Generate code with df | New interactive sheet

```
#Dataset Information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Genre                   200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
#Statistical Summary
df.describe()
```

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

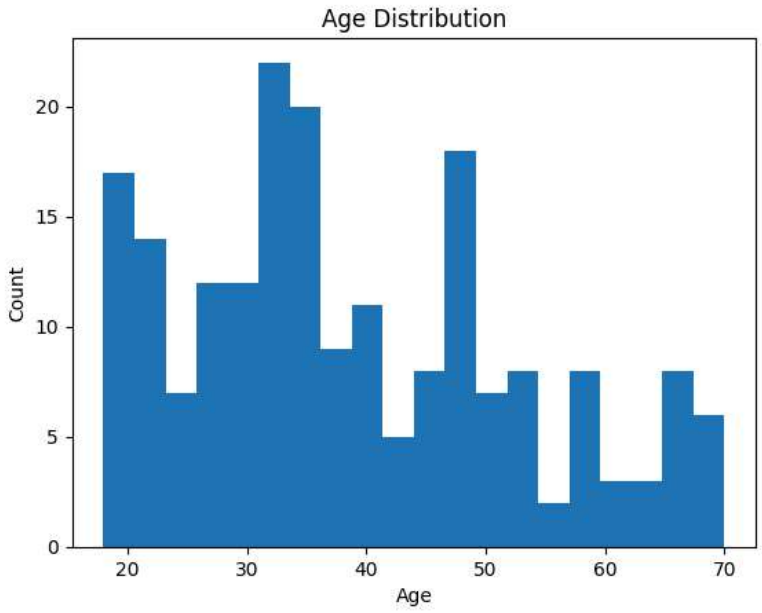
```
#Null Value Check
df.isnull().sum()
```

	0
CustomerID	0
Genre	0
Age	0
Annual Income (k$)	0
Spending Score (1-100)	0

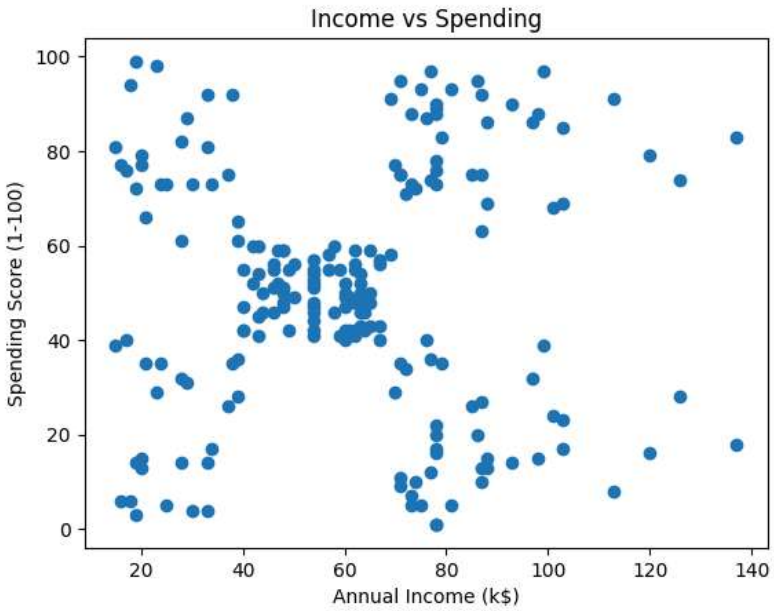
dtype: int64

Data Visualization

```
#Age Distribution
plt.hist(df['Age'], bins=20)
plt.title("Age Distribution")
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()
```




```
#Income vs Spending
plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'])
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.title("Income vs Spending")
plt.show()
```



K-Means Clustering (Unsupervised Learning)

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

```
X_cluster = df[['Annual Income (k$)', 'Spending Score (1-100)']]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_cluster)

kmeans = KMeans(n_clusters=5, random_state=42)
df['KMeans_Cluster'] = kmeans.fit_predict(X_scaled)
```

High & Low Spender Clusters

```
cluster_mean = df.groupby('KMeans_Cluster')['Spending Score (1-100)'].mean()

high_cluster = cluster_mean.idxmax()
low_cluster = cluster_mean.idxmin()

kmeans_counts = {
    "High Spenders": (df['KMeans_Cluster'] == high_cluster).sum(),
    "Low Spenders": (df['KMeans_Cluster'] == low_cluster).sum()
}

kmeans_counts
```

```
{'High Spenders': np.int64(39), 'Low Spenders': np.int64(35)}
```

```
#Silhouette Score
score = silhouette_score(X_scaled, df['KMeans_Cluster'])
print("Silhouette Score:", score)
```

```
Silhouette Score: 0.5546571631111091
```

```
#Highest & Lowest Spending Customers
max_spender = df.loc[df['Spending Score (1-100)'].idxmax()]
min_spender = df.loc[df['Spending Score (1-100)'].idxmin()]

print("Highest Spender")
print(max_spender[['CustomerID','Annual Income (k$)','Spending Score (1-100)']])

print("\nLowest Spender")
print(min_spender[['CustomerID','Annual Income (k$)','Spending Score (1-100)']])
```

```
Highest Spender
CustomerID               12
Annual Income (k$)       19
Spending Score (1-100)   99
Name: 11, dtype: object

Lowest Spender
CustomerID               157
Annual Income (k$)        78
Spending Score (1-100)     1
Name: 156, dtype: object
```

Linear Regression

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
X = df[['Annual Income (k$)', 'Age']]
y = df['Spending Score (1-100)']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

lr = LinearRegression()
lr.fit(X_train, y_train)

y_pred = lr.predict(X_test)

print("MSE:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
MSE: 483.55682175408344
R² Score: 0.01963177813218009
```

```
# Predict spending for all customers
df['LR_Predicted_Spending'] = lr.predict(df[['Annual Income (k$)', 'Age']])

print("LINEAR REGRESSION - Highest Predicted Spender")
print(df.loc[df['LR_Predicted_Spending'].idxmax()]
      [['CustomerID','Annual Income (k$)','LR_Predicted_Spending']])

print("\nLINEAR REGRESSION - Lowest Predicted Spender")
print(df.loc[df['LR_Predicted_Spending'].idxmin()]
      [['CustomerID','Annual Income (k$)','LR_Predicted_Spending']])
```

```
LINEAR REGRESSION - Highest Predicted Spender
CustomerID                    163
Annual Income (k$)             81
LR_Predicted_Spending    64.577057
Name: 162, dtype: object

LINEAR REGRESSION - Lowest Predicted Spender
CustomerID                     61
Annual Income (k$)             46
LR_Predicted_Spending    32.690629
Name: 60, dtype: object
```

```
#Classification Setup (High vs Low Spenders)
df['Spender_Class'] = (df['Spending Score (1-100)'] >= 50).astype(int)
```

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

```
X = df[['Annual Income (k$)', 'Age']]
y = df['Spender_Class']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

dt = DecisionTreeClassifier(max_depth=4, random_state=42)
dt.fit(X_train, y_train)

y_pred = dt.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred))
```

```
Decision Tree Accuracy: 0.675
```

```
df['DT_Predicted_Class'] = dt.predict(df[['Annual Income (k$)', 'Age']])

dt_high = df[df['DT_Predicted_Class'] == 1]
dt_low = df[df['DT_Predicted_Class'] == 0]

print("DECISION TREE - Highest Spender")
print(dt_high.loc[dt_high['Spending Score (1-100)'].idxmax()]
      [['CustomerID','Annual Income (k$)','Spending Score (1-100)']])

print("\nDECISION TREE - Lowest Spender")
print(dt_low.loc[dt_low['Spending Score (1-100)'].idxmin()]
      [['CustomerID','Annual Income (k$)','Spending Score (1-100)']])
```

```
DECISION TREE - Highest Spender
CustomerID               12
Annual Income (k$)       19
Spending Score (1-100)   99
Name: 11, dtype: object

DECISION TREE - Lowest Spender
CustomerID                9
Annual Income (k$)       19
Spending Score (1-100)    3
Name: 8, dtype: object
```

K-Nearest Neighbors (KNN)

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)

y_pred = knn.predict(X_test_scaled)
print("KNN Accuracy:", accuracy_score(y_test, y_pred))
```

```
KNN Accuracy: 0.65
```

```
df['KNN_Predicted_Class'] = knn.predict(
    scaler.transform(df[['Annual Income (k$)', 'Age']])
)

knn_high = df[df['KNN_Predicted_Class'] == 1]
knn_low = df[df['KNN_Predicted_Class'] == 0]

print("KNN - Highest Spender")
print(knn_high.loc[knn_high['Spending Score (1-100)'].idxmax()]
      [['CustomerID','Annual Income (k$)','Spending Score (1-100)']])

print("\nKNN - Lowest Spender")
print(knn_low.loc[knn_low['Spending Score (1-100)'].idxmin()]
      [['CustomerID','Annual Income (k$)','Spending Score (1-100)']])
```

```
KNN - Highest Spender
CustomerID               20
Annual Income (k$)       23
Spending Score (1-100)   98
Name: 19, dtype: object

KNN - Lowest Spender
CustomerID                9
Annual Income (k$)       19
Spending Score (1-100)    3
Name: 8, dtype: object
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
```

```
rf = RandomForestClassifier(
    n_estimators=200,
    max_depth=6,
    min_samples_split=4,
    random_state=42)

rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)
```

```
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred))
```

```
Random Forest Accuracy: 0.675
```

```
df['RF_Predicted_Class'] = rf.predict(df[['Annual Income (k$)', 'Age']])

rf_high = df[df['RF_Predicted_Class'] == 1]
rf_low = df[df['RF_Predicted_Class'] == 0]

print("RANDOM FOREST - Highest Spender")
print(rf_high.loc[rf_high['Spending Score (1-100)'].idxmax()]
      [['CustomerID','Annual Income (k$)','Spending Score (1-100)']])

print("\nRANDOM FOREST - Lowest Spender")
print(rf_low.loc[rf_low['Spending Score (1-100)'].idxmin()]
      [['CustomerID','Annual Income (k$)','Spending Score (1-100)']])
```

```
RANDOM FOREST - Highest Spender
CustomerID               12
Annual Income (k$)       19
Spending Score (1-100)   99
Name: 11, dtype: object

RANDOM FOREST - Lowest Spender
CustomerID                9
Annual Income (k$)       19
Spending Score (1-100)    3
Name: 8, dtype: object
```

printing best algorithm

```
from sklearn.metrics import accuracy_score

#Decision Tree Accuracy
dt_accuracy = accuracy_score(y_test, dt.predict(X_test))

#KNN Accuracy
knn_accuracy = accuracy_score(y_test, knn.predict(X_test_scaled))

#Random Forest Accuracy
rf_accuracy = accuracy_score(y_test, rf.predict(X_test))

#linear regression
r2 = r2_score(y_test, lr.predict(X_test))


# Store Scores
scores = {
    "KMeans": score,         # Silhouette Score
    "Decision Tree": dt_accuracy,
    "Linear Regression":r2,
    "KNN": knn_accuracy,
    "Random Forest": rf_accuracy
}

#Find Best Algorithm
best_algorithm = max(scores, key=scores.get)

#Print Result
```

```
print(" BEST ALGORITHM ")
print(f"Algorithm: {best_algorithm}")
print(f"Score: {scores[best_algorithm]:.3f}")
```

```
 BEST ALGORITHM
Algorithm: Decision Tree
Score: 0.675
```

Adding new data to predict

```
# New customer
new_customer = pd.DataFrame({
    'Annual Income (k$)': [30],
    'Age': [75]
})

prediction = dt.predict(new_customer)[0]

print("HIGH SPENDER" if prediction == 1 else "LOW SPENDER")
```

```
LOW SPENDER
```