Data Science QUESTION 2

You'll be asked to do some analysis and modeling tasks on a dataset created.
The dataset concerns a video gaming company that has information on its customers and would like to gain more insights on what drives their customers to play for longer hours.

The following tables are provided:

In [302…

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

In [303…

```
df_districts_house_prices = pd.read_csv("df_districts_house_prices.csv")
df_person_data = pd.read_csv("df_person_data.csv")
df_person_district = pd.read_csv("df_person_district.csv")
```

df_person_district

includes information about each customer (person) and the district they live in

- person_id: the person id
- district: the name of the district

In [304…

```
df_person_district
```

Out[304]:

	person_id	district
0	50c4c7e2-89a6-440b-a8e3-c44aa2c6150e	Metdunstone
1	16f3bafb-9556-434e-adab-cb02f41fe32a	Tashnerspool
2	0fa17eee-7214-4609-97fe-dd3093601800	Tashnerspool
3	8db4ca66-dfb2-43f2-9c22-aa861dd0d218	Ulven
4	51fed64a-375e-417f-94e4-4d27c368ea44	Red Onvey
...	...	...
69995	8cbea1e2-3257-4db5-ae59-84faa48cfde2	Bluffssel
69996	e099ace5-a760-4362-a9ea-ae8cee590b86	Highnantmar
69997	e9b0604e-ec84-4704-8e75-eabfd59ac4fb	Tashnerspool
69998	e20f8ec5-0ddf-4674-ac1f-d6280b6640ab	Highnantmar
69999	4d86ea85-2eaf-457a-bdba-73ff21830588	San Readma

70000 rows × 2 columns

df_districts_house_prices

includes information about each district and the prices of the houses in the district

- district: the name of the district
- house_price: the price of the house
- house_number: the house number in the district

In [305…

```
df_districts_house_prices
```

Out[305]:

	district	house_price	house_number
0	Celowsgan	160652.0	1
1	Celowsgan	159219.0	2
2	Celowsgan	161543.0	3
3	Celowsgan	158944.0	4
4	Celowsgan	164121.0	5
...	...	...	...
1358	El Willong	932441.0	30
1359	El Willong	890190.0	31
1360	El Willong	892096.0	32
1361	El Willong	935117.0	33
1362	El Willong	953480.0	34

1363 rows × 3 columns

df_person_data

includes personal information about each of the customers and relevant information to their video gaming habits

- person_id: identifier for a person
- age: the age of the person
- n_kids: the number of kids this person has
- n_vg: the number of video games the person owns
- n_con: the number of video game consols the person owns
- n_presub: the number of premium subscription the person owns
- n_hours_playing: the total number of hours this person play per month

In [306…

```
df_person_data
```

Out[306]:

	person_id	age	n_kids	n_vg	n_con	n_presub	n_hours_playing
0	50c4c7e2-89a6-440b-a8e3-c44aa2c6150e	14.0	0	0	0	0	18.422745
1	16f3bafb-9556-434e-adab-cb02f41fe32a	18.0	0	2	0	0	20.693273
2	0fa17eee-7214-4609-97fe-dd3093601800	28.0	0	3	0	0	22.412490
3	8db4ca66-dfb2-43f2-9c22-aa861dd0d218	20.0	1	72	0	0	299.187025
4	51fed64a-375e-417f-94e4-4d27c368ea44	32.0	1	58	1	3	20.367141
...	...	...	...	...	...	...	...
69995	8cbea1e2-3257-4db5-ae59-84faa48cfde2	32.0	4	62	1	2	21.378288
69996	e099ace5-a760-4362-a9ea-ae8cee590b86	36.0	0	53	1	1	3.707476
69997	e9b0604e-ec84-4704-8e75-eabfd59ac4fb	19.0	0	1	1	0	23.809075
69998	e20f8ec5-0ddf-4674-ac1f-d6280b6640ab	31.0	2	49	3	2	15.708397
69999	4d86ea85-2eaf-457a-bdba-73ff21830588	20.0	0	3	0	2	30.796314

70000 rows × 7 columns

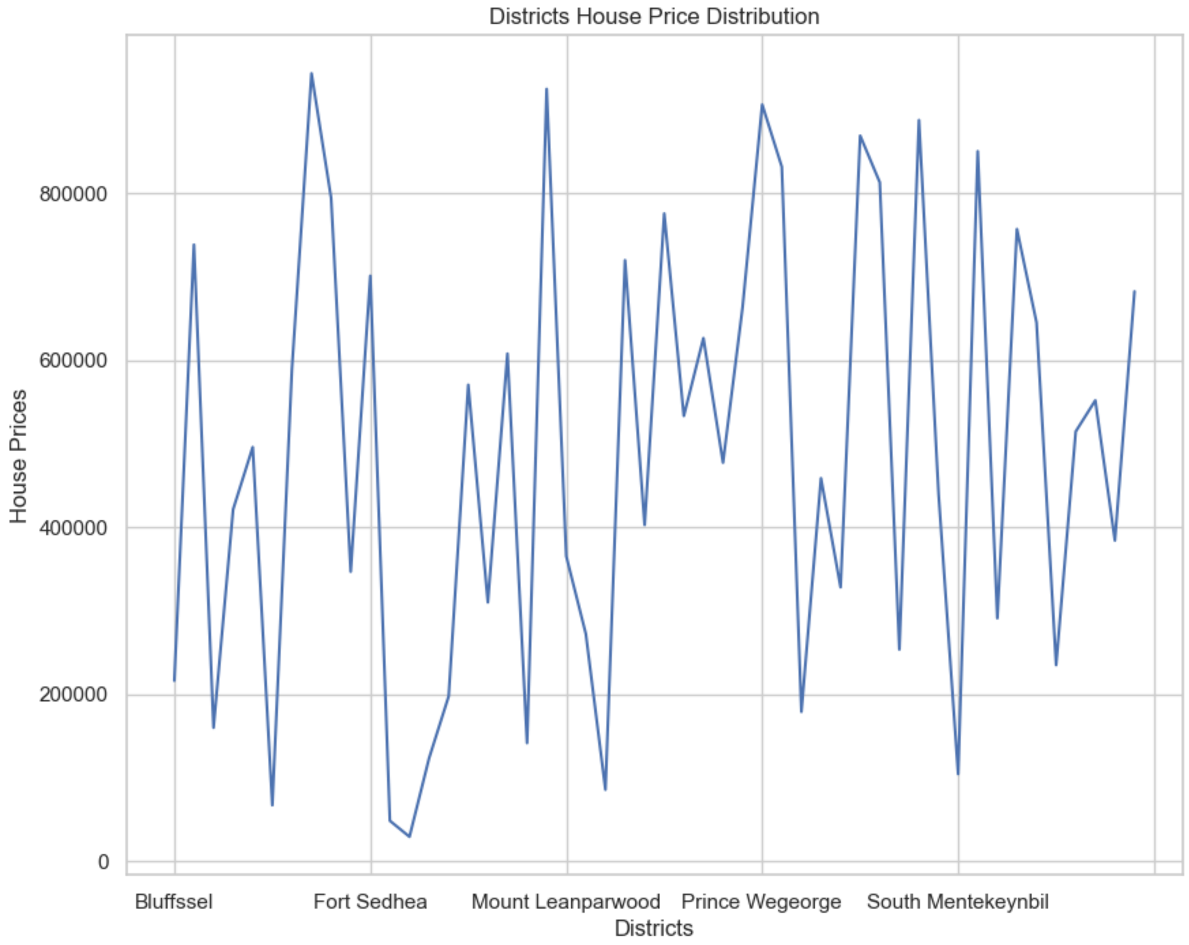
Quick EDA

The goals of these questions is to evaluate your plotting, data mangling, and plot interpretation skills.
Use whatever libraries you are comfortable with.
Code clarity and cleanliness are also highly valuable.

1. Plot each district's house prices distribution with marking the mean by a vertical line

In [331…

```
plt.figure(figsize=(10,8))
df_districts_house_prices.groupby(["district"]).house_price.mean().plot()
plt.xlabel('Districts')
plt.ylabel('House Prices')
plt.title('Districts House Price Distribution')
plt.show()
```



2. Combine all of the three data sources into one table to use in further analysis.

In [332…

```
combined_3_dfs = pd.concat([df_person_data, df_person_district, df_districts_house_
```

In [333…

```
combined_3_dfs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   person_id        70000 non-null  object 
 1   age              69990 non-null  float64
 2   n_kids           70000 non-null  int64  
 3   n_vg             70000 non-null  int64  
 4   n_con            70000 non-null  int64  
 5   n_presub         70000 non-null  int64  
 6   n_hours_playing  70000 non-null  float64
 7   person_id        70000 non-null  object 
 8   district         70000 non-null  object 
 9   district         1363 non-null   object 
 10  house_price      1352 non-null   float64
 11  house_number     1363 non-null   float64
dtypes: float64(4), int64(4), object(4)
memory usage: 6.4+ MB
```

In [334…

```
combined_3_dfs.isnull().sum()
```

Out[334]:

```
person_id               0
age                    10
n_kids                  0
n_vg                    0
n_con                   0
n_presub                0
n_hours_playing         0
person_id               0
district                0
district            68637
house_price         68648
house_number        68637
dtype: int64
```

In [335…

```
mean_age = round(combined_3_dfs['age'].mean(),1)
print("Mean Age = ", mean_age)
combined_3_dfs['age'].fillna(mean_age, inplace=True)
```

```
Mean Age =  25.0
```

In [336…

```
combined_3_dfs.isnull().sum()
```

Out[336]:

```
person_id               0
age                     0
n_kids                  0
n_vg                    0
n_con                   0
n_presub                0
n_hours_playing         0
person_id               0
district                0
district            68637
house_price         68648
house_number        68637
dtype: int64
```

In [337…

```
combined_3_dfs.head()
```

Out[337]:

	person_id	age	n_kids	n_vg	n_con	n_presub	n_hours_playing	person_id
0	50c4c7e2-89a6-440b-a8e3-c44aa2c6150e	14.0	0	0	0	0	18.422745	50c4c7e2-89a6-440b-a8e3-c44aa2c6150e
1	16f3bafb-9556-434e-adab-cb02f41fe32a	18.0	0	2	0	0	20.693273	16f3bafb-9556-434e-adab-cb02f41fe32a
2	0fa17eee-7214-4609-97fe-dd3093601800	28.0	0	3	0	0	22.412490	0fa17eee-7214-4609-97fe-dd3093601800
3	8db4ca66-dfb2-43f2-9c22-aa861dd0d218	20.0	1	72	0	0	299.187025	8db4ca66-dfb2-43f2-9c22-aa861dd0d218
4	51fed64a-375e-417f-94e4-4d27c368ea44	32.0	1	58	1	3	20.367141	51fed64a-375e-417f-94e4-4d27c368ea44

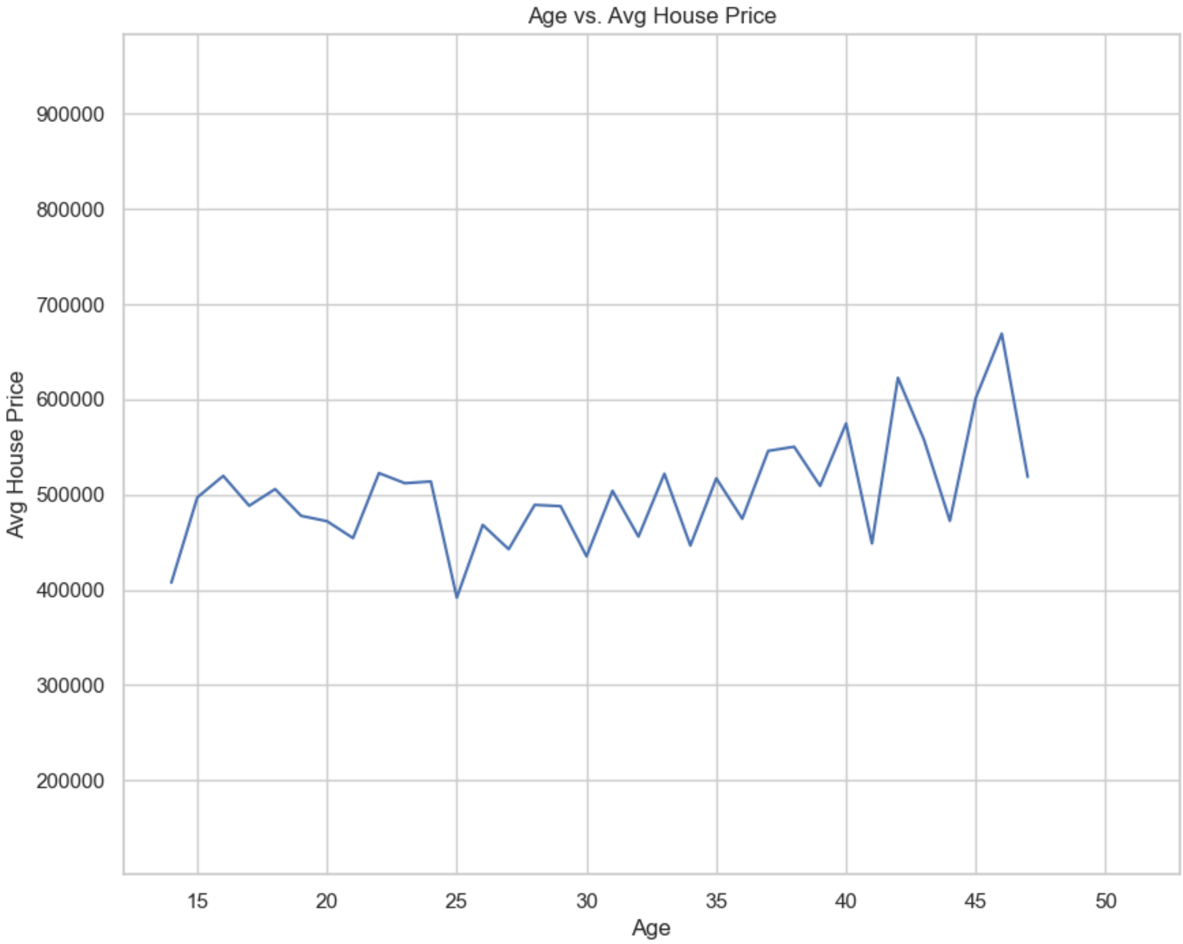
3. Plot age vs avg house price. What does this plot tell you? (younger people live in more expensive districts)

In [339…

```
plt.figure(figsize=(10,8))
mean_house_price = combined_3_dfs.groupby('age')['house_price'].mean()

plt.plot(mean_house_price.index, mean_house_price.values)
plt.xlabel('Age')
plt.ylabel('Avg House Price')
plt.title('Age vs. Avg House Price')
plt.show()

# With increasing age the avg price of house is increasing that show older people l
```



Probability and Statistics:

The goal of these questions is to test your ability to answer probability and stat questions with code.
Use whatever libraries you are comfortable with.
Code clarity and cleanliness are also highly valuable.

1. What's the probability of a customer having 2 kids

In [340…

```
customers_having_two_kids = combined_3_dfs[combined_3_dfs['n_kids'] == 2]
probability_2_kids = len(customers_having_two_kids) / len(combined_3_dfs)

print("Customer having Two kids Probability = ", probability_2_kids * 100, "%")
```

```
Customer having Two kids Probability =  10.017142857142858 %
```

2. What's the probability of a customer owning more than 10 video games given that they have less than 2 kids

In [341…

```
customers_having_less_than_two_kids = combined_3_dfs[combined_3_dfs['n_kids'] < 2]
print("Customers having less then two kids = ", len(customers_having_less_than_two_
customers_having_more_than_10_videogames = customers_having_less_than_two_kids[cust
print("Customers having more then ten video games = ", len(customers_having_more_th
probability = len(customers_having_more_than_10_videogames) / len(customers_having_
print("Customers having less then 2 kids and more then 10 video games Probability =
```

```
Customers having less then two kids =  50100
Customers having more then ten video games =  20132
Customers having less then 2 kids and more then 10 video games Probability =  40.18
363273453094 %
```

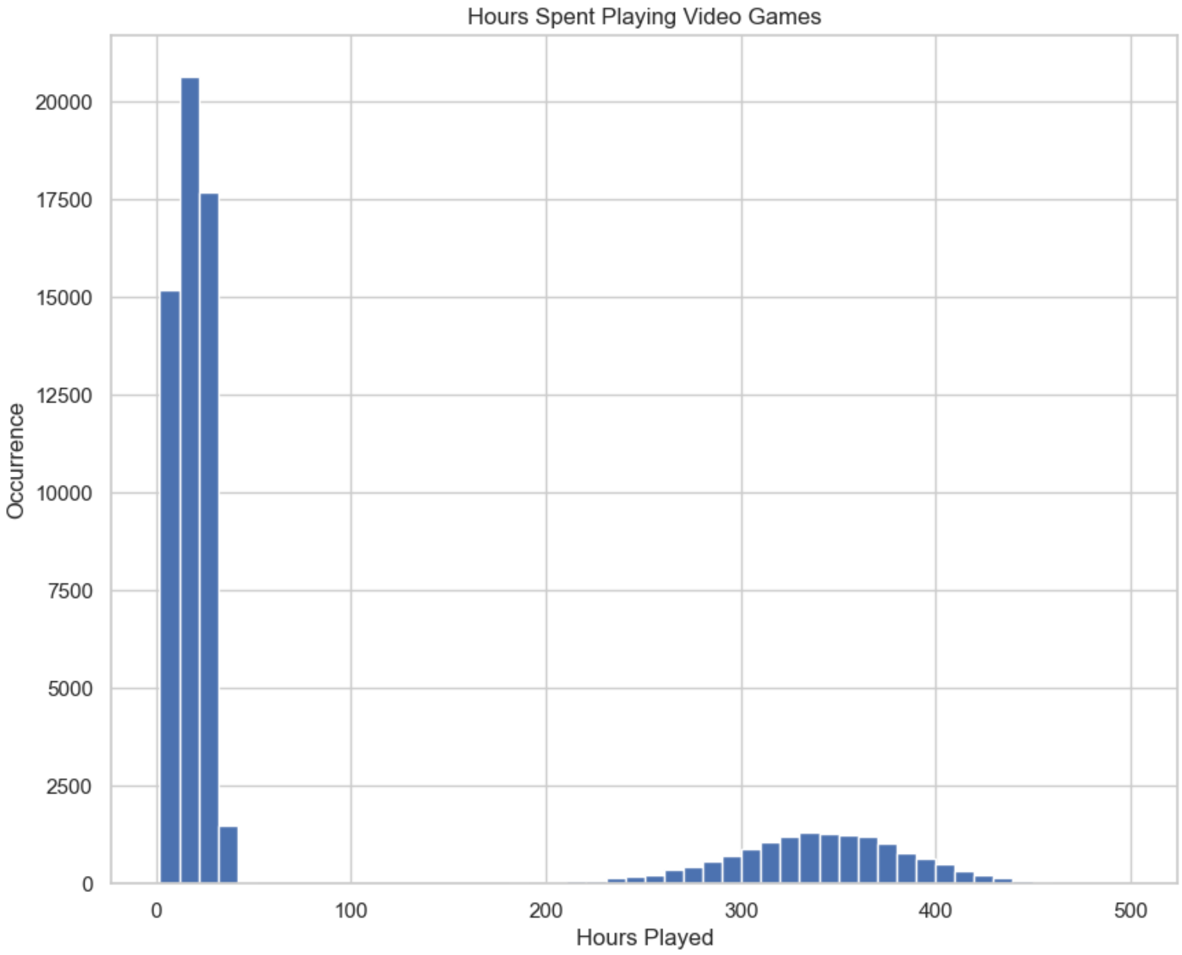
3. Plot the distribution of the number of hours played by customers n_hours_playing . Briefly explain what you understand now about the customers' playing hours.

In [342…

```
plt.figure(figsize=(10,8))
customers_played_less_then_50_hrs = combined_3_dfs[combined_3_dfs['n_hours_playing'
print("Customers Played Less then 50 hrs = ", len(customers_played_less_then_50_hrs
print("Total Customers = ", len(combined_3_dfs))
print("Probability playing less then 50 hrs = ",len(customers_played_less_then_50_h
plt.hist(combined_3_dfs['n_hours_playing'], bins=50)
plt.xlabel('Hours Played')
plt.ylabel('Occurrence')
plt.title('Hours Spent Playing Video Games')
plt.show()

# According to given data most customers playing hours are less then 50.
```

```
Customers Played Less then 50 hrs =  54997
Total Customers =  70000
Probability playing less then 50 hrs =  78.56714285714285 %
```



As you can see, most customers have a number of hours less than 50 so let's go ahead and remove any values less than 50.
The distribution of the remaining values look like a normal distribution.

4. Estimate the parameters (mean and std deviation) of this normal distribution computationally. \

(Bonus: plot the estimated normal distribution on top of the distribution of `n_hours_playing` after removing values < 50)

In [343…

```
std_deviation_playing_hours = combined_3_dfs['n_hours_playing'].std()

mean_playing_hours = combined_3_dfs['n_hours_playing'].mean()

print("Mean = ", mean_playing_hours)
print("Standard Deviation = ", std_deviation_playing_hours)
```

```
Mean =  86.58507315764899
Standard Deviation =  133.66556897706656
```

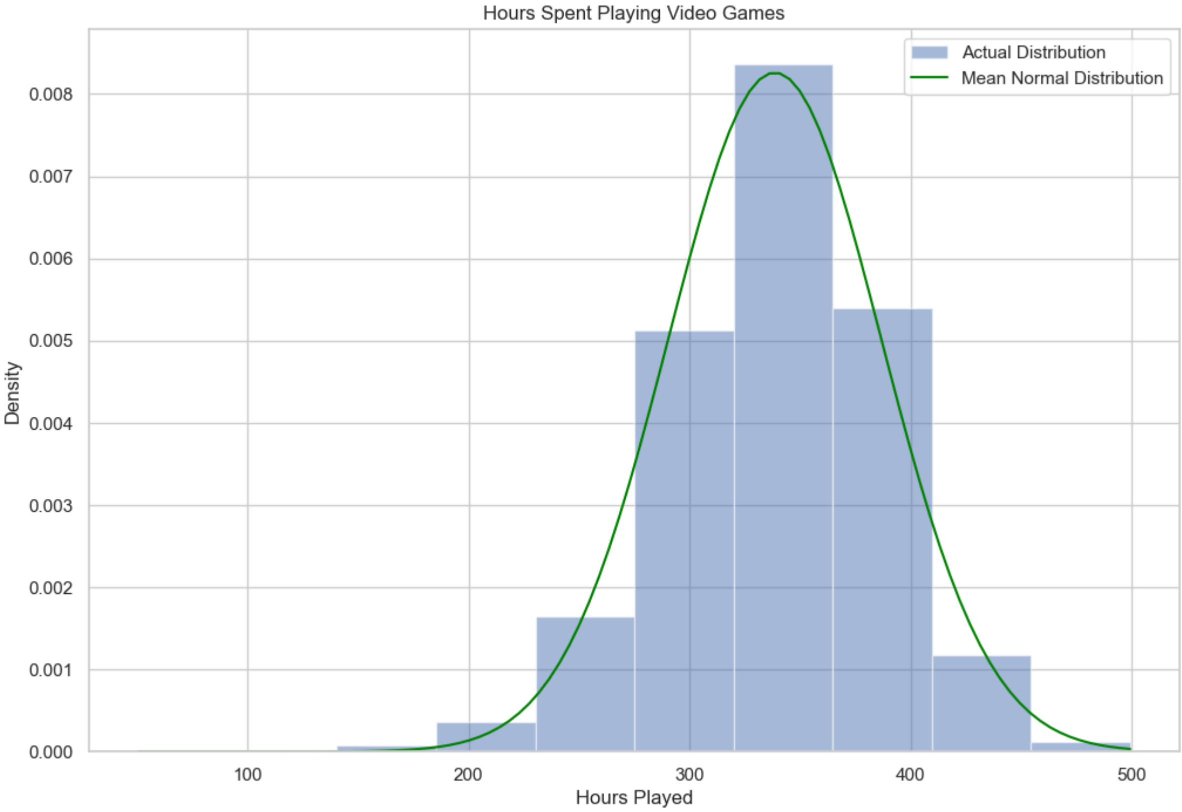
In [344…

```
more_then_fifty_hours = combined_3_dfs[combined_3_dfs['n_hours_playing'] >= 50]['n_
print("Customers with more then 50 playing hours = ", len(more_then_fifty_hours))
std_deviation_playing_hours = more_then_fifty_hours.std()
mean_playing_hours = more_then_fifty_hours.mean()

x_axis = np.linspace(more_then_fifty_hours.min(), more_then_fifty_hours.max(), 100)
y_axis = (1 / (std_deviation_playing_hours * np.sqrt(2 * np.pi))) * np.exp(-(x_axis

plt.figure(figsize=(12,8))
plt.hist(more_then_fifty_hours, bins=10, density=True, alpha=0.5, label='Actual Dis
plt.plot(x_axis, y_axis, color='green', label='Mean Normal Distribution')
plt.xlabel('Hours Played')
plt.ylabel('Density')
plt.title('Hours Spent Playing Video Games')
plt.legend()
plt.show()
```

```
Customers with more then 50 playing hours =  15003
```



Modeling

The goal of this is to showcase your experimentation and model comparison process. The performance of models is not as important as how you compare them and evaluate them against each other.

Use whatever libraries you are comfortable with.
Code clarity and cleanliness are also highly valuable.

Apply Different regression model

Apply two classffier Model for the data of classfication (as per your choice)

1. Model the number of hours played for each customer. Show your experimentation with failed and successful models.

2. Evaluate each experiment using appropriate metrics, cross validation, and plots. Show the predicted vs actual plot.

3. Choose the best model for the data

In [345…

```
combined_3_dfs.head()
```

Out[345]:

	person_id	age	n_kids	n_vg	n_con	n_presub	n_hours_playing	person_id
0	50c4c7e2-89a6-440b-a8e3-c44aa2c6150e	14.0	0	0	0	0	18.422745	50c4c7e2-89a6-440b-a8e3-c44aa2c6150e
1	16f3bafb-9556-434e-adab-cb02f41fe32a	18.0	0	2	0	0	20.693273	16f3bafb-9556-434e-adab-cb02f41fe32a
2	0fa17eee-7214-4609-97fe-dd3093601800	28.0	0	3	0	0	22.412490	0fa17eee-7214-4609-97fe-dd3093601800
3	8db4ca66-dfb2-43f2-9c22-aa861dd0d218	20.0	1	72	0	0	299.187025	8db4ca66-dfb2-43f2-9c22-aa861dd0d218
4	51fed64a-375e-417f-94e4-4d27c368ea44	32.0	1	58	1	3	20.367141	51fed64a-375e-417f-94e4-4d27c368ea44

In [346…

```
combined_3_dfs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   person_id        70000 non-null  object 
 1   age              70000 non-null  float64
 2   n_kids           70000 non-null  int64  
 3   n_vg             70000 non-null  int64  
 4   n_con            70000 non-null  int64  
 5   n_presub         70000 non-null  int64  
 6   n_hours_playing  70000 non-null  float64
 7   person_id        70000 non-null  object 
 8   district         70000 non-null  object 
 9   district         1363 non-null   object 
 10  house_price      1352 non-null   float64
 11  house_number     1363 non-null   float64
dtypes: float64(4), int64(4), object(4)
memory usage: 6.4+ MB
```

In [347…

```
combined_3_dfs.drop(["person_id", "district", "house_price", "house_number"], axis=
```

In [348…

```
combined_3_dfs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   age              70000 non-null  float64
 1   n_kids           70000 non-null  int64  
 2   n_vg             70000 non-null  int64  
 3   n_con            70000 non-null  int64  
 4   n_presub         70000 non-null  int64  
 5   n_hours_playing  70000 non-null  float64
dtypes: float64(2), int64(4)
memory usage: 3.2 MB
```

In [349…

```
input_data = combined_3_dfs.drop(columns=['n_hours_playing'], axis=1)
output_data = combined_3_dfs['n_hours_playing']
```

In [350…

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(input_data, output_data, test_s
```

Linear Regression:

In [351…

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_pr
from sklearn.metrics import mean_squared_error, r2_score

def apply_model(model, input_data, output_data):
    x_train, x_test, y_train, y_test = train_test_split(input_data, output_data, te
    model.fit(x_train, y_train)
    model_accuracy = model.score(x_test, y_test) * 100
    print("Model Accuracy = ", model_accuracy, "%")
    y_cvp = cross_val_predict(model, input_data, output_data, cv=10)
    mse = mean_squared_error(output_data, y_cvp)
    r2 = r2_score(output_data, y_cvp)
    print("Modal Mean Squared Error = ", mse)
    print("Modal R2 Score = ", r2)
    plt.scatter(output_data, y_cvp, alpha=0.5)
    plt.plot([output_data.min(), output_data.max()], [output_data.min(), output_dat
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title('Predicted vs Actual')
    plt.show()
```

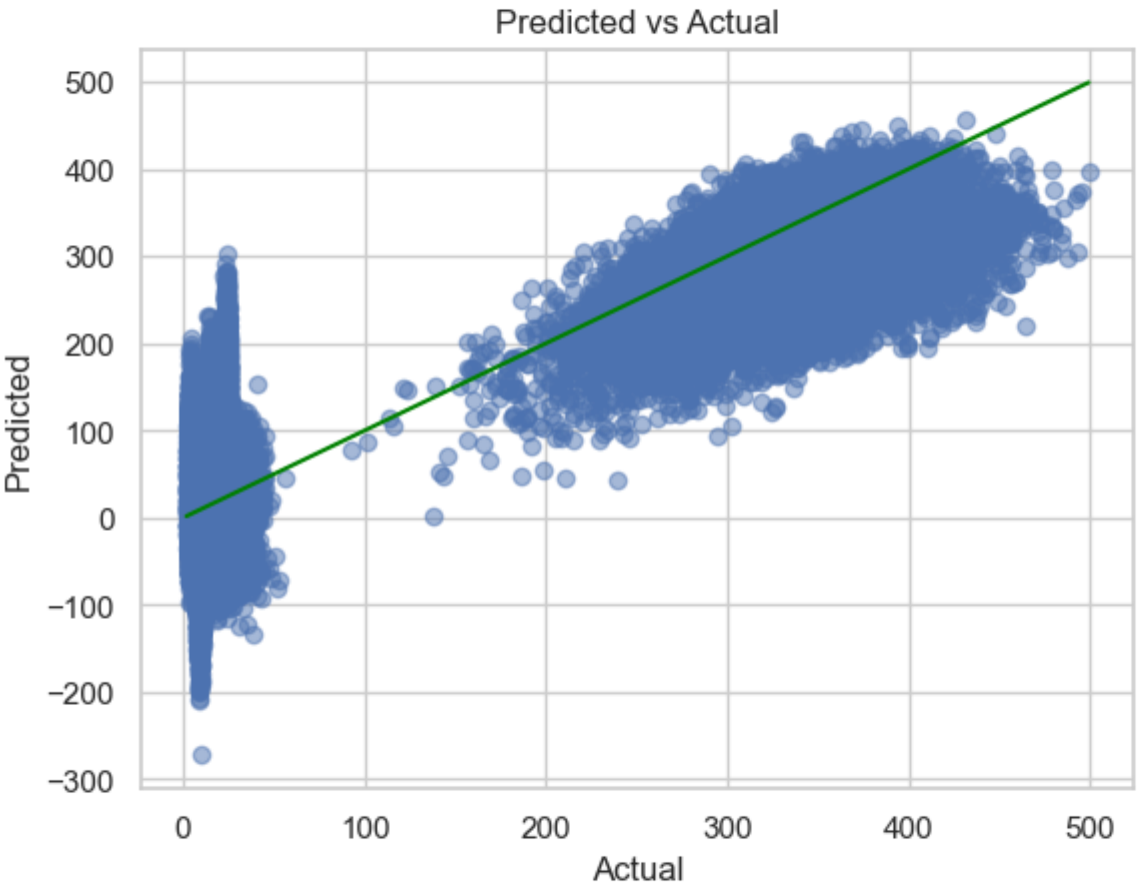
In [352…

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

In [353…

```
apply_model(LinearRegression(), input_data, output_data)
```

```
Model Accuracy =  81.30895899112139 %
Modal Mean Squared Error =  3309.007197370168
Modal R2 Score =  0.8147898373029405
```



Decision Tree Regressor:

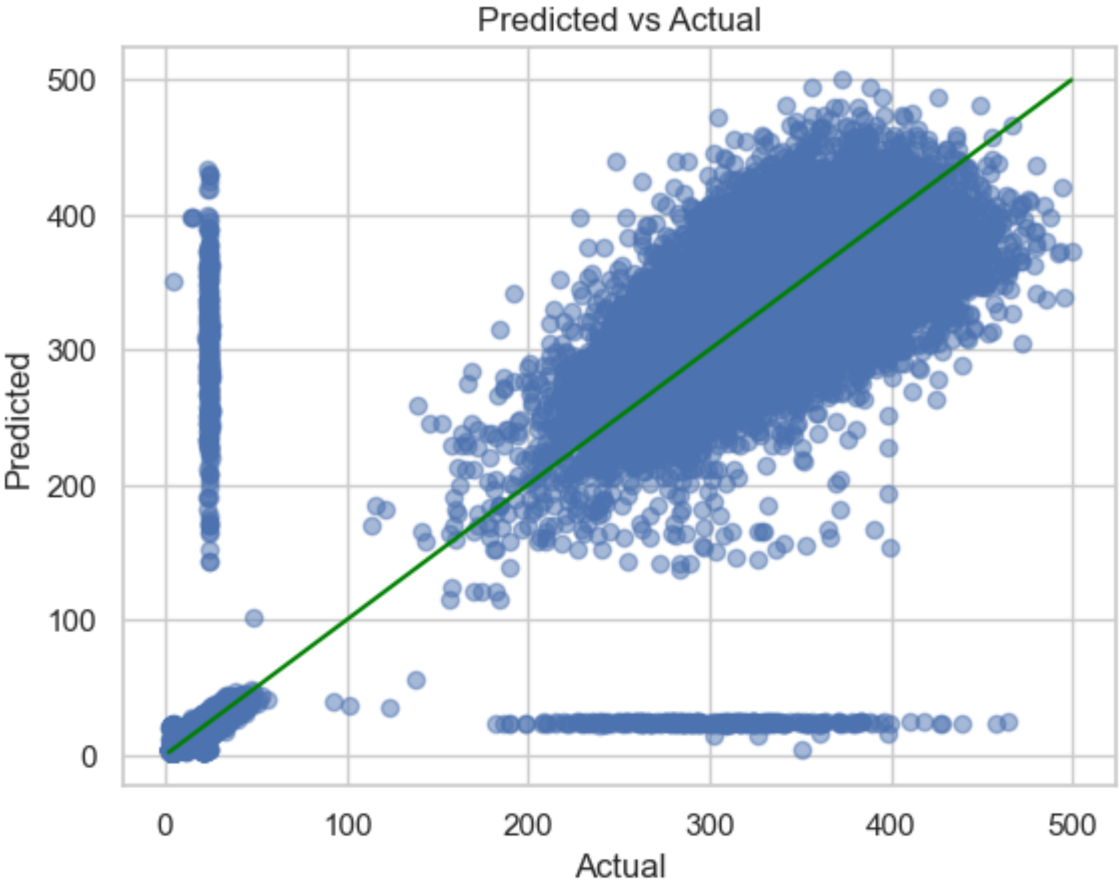
In [354…

```
from sklearn.tree import DecisionTreeRegressor
```

In [355…

```
apply_model(DecisionTreeRegressor(), input_data, output_data)
```

```
Model Accuracy =  94.33369985759104 %
Modal Mean Squared Error =  953.1979684166264
Modal R2 Score =  0.9466480607980373
```


Predicted vs Actual
Predicted
Actual