In [1]:
```
# importing all the libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

Bottle Dataset

(linear regression model)

In [2]:
```
# reading the file
df=pd.read_csv(r"C:\Users\91756\Documents\python\bottle.csv")
df
```

```
C:\Users\91756\AppData\Local\Temp\ipykernel_23128\1390274385.py:2: DtypeWarn
ing: Columns (47,73) have mixed types. Specify dtype option on import or set
low_memory=False.
  df=pd.read_csv(r"C:\Users\91756\Documents\python\bottle.csv")
```

Out[2]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2S
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.500	33.4400	NaN	25.64900	N
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.460	33.4400	NaN	25.65600	N
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.460	33.4370	NaN	25.65400	N
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.450	33.4200	NaN	25.64300	N
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.450	33.4210	NaN	25.64300	N
...	...	...	...	...	...	...	...	...	...	
864858	34404	864859	093.4 026.4	20-1611SR-MX-310-2239-09340264-0000A-7	0	18.744	33.4083	5.805	23.87055	108
864859	34404	864860	093.4 026.4	20-1611SR-MX-310-2239-09340264-0002A-3	2	18.744	33.4083	5.805	23.87072	108
864860	34404	864861	093.4 026.4	20-1611SR-MX-310-2239-09340264-0005A-3	5	18.692	33.4150	5.796	23.88911	108
864861	34404	864862	093.4 026.4	20-1611SR-MX-310-2239-09340264-0010A-3	10	18.161	33.4062	5.816	24.01426	107

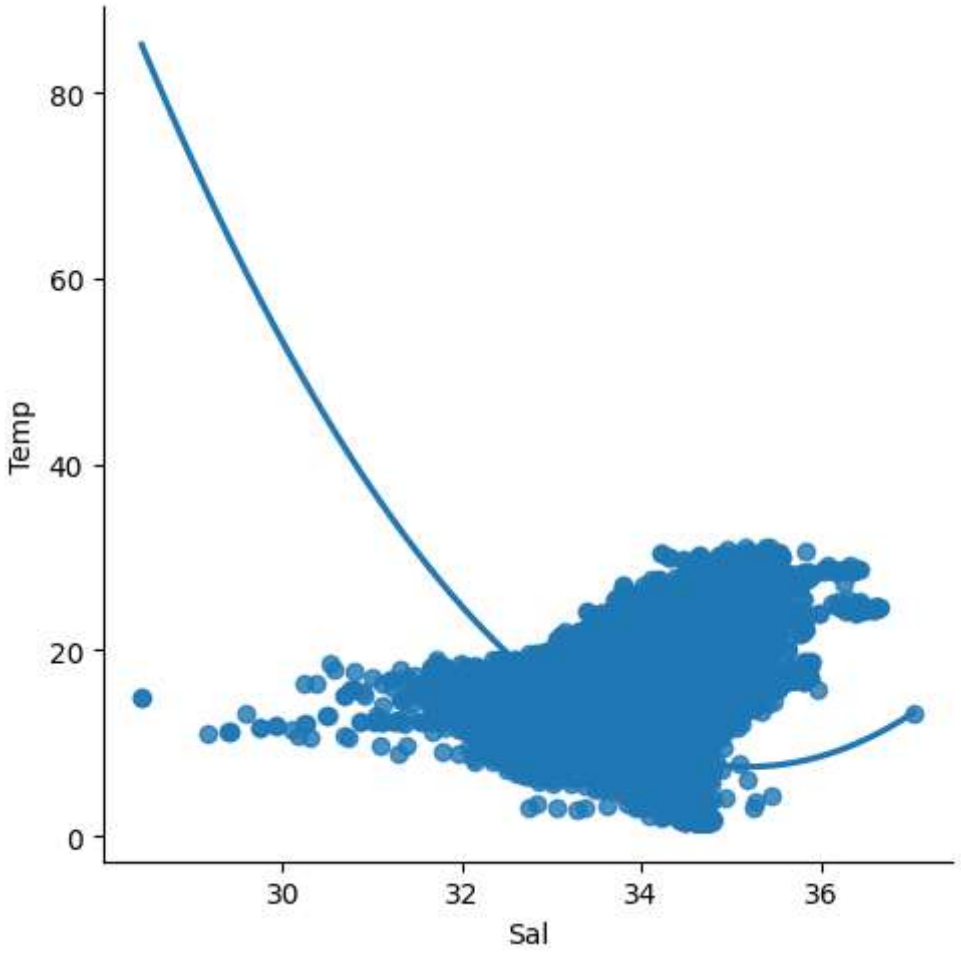
	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2S
864862	34404	864863	093.4 026.4	20-1611SR-MX-310-2239-09340264-0015A-3	15	17.533	33.3880	5.774	24.15297	105

864863 rows × 74 columns

In [3]:
```
df = df[['Salnty', 'T_degC']]
df.columns=['Sal','Temp']
```

In [4]:
```
# step 3: Exploring the data scatter _plotting the data scatter
sns.lmplot(x="Sal", y="Temp", data=df, order=2, ci= None)
```

Out[4]: <seaborn.axisgrid.FacetGrid at 0x1f73b904f70>



In [5]:
```
df.describe()
```

Out[5]:

	Sal	Temp
count	817509.000000	853900.000000
mean	33.840350	10.799677
std	0.461843	4.243825
min	28.431000	1.440000
25%	33.488000	7.680000
50%	33.863000	10.060000
75%	34.196900	13.880000
max	37.034000	31.140000

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864863 entries, 0 to 864862
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Sal     817509 non-null  float64
 1   Temp    853900 non-null  float64
dtypes: float64(2)
memory usage: 13.2 MB
```

In [7]:
```
# step-4: Data cleaning- Eliminating NaN or missing input numbers
df.fillna(method='ffill', inplace=True)
```

```
C:\Users\91756\AppData\Local\Temp\ipykernel_23128\1327383682.py:2: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df.fillna(method='ffill', inplace=True)
```

In [8]:
```
# step-5: Training our Model
X = np.array(df['Sal']).reshape(-1,1)
y = np.array(df['Temp']).reshape(-1,1)
# Separating the data into independent and dependent variables and converting
# Now each dataframe contains only one column
```

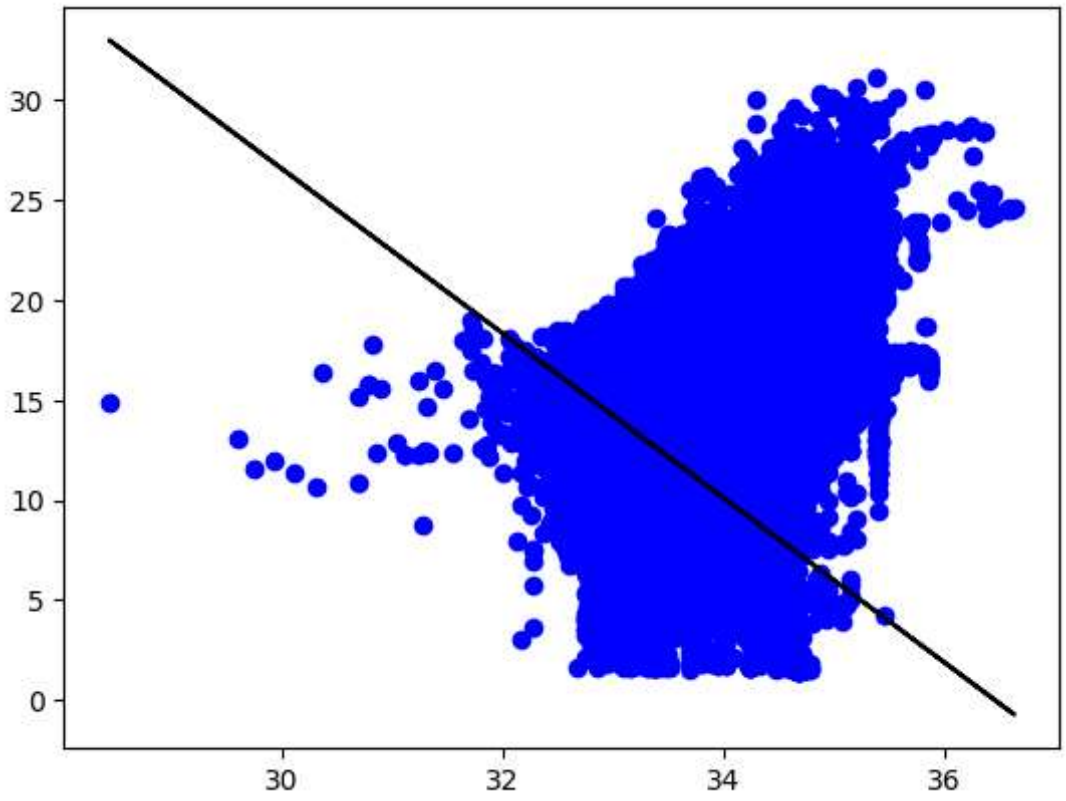
In [9]:
```
df.dropna(inplace = True)
# Dropping any rows with Nan values
```

```
C:\Users\91756\AppData\Local\Temp\ipykernel_23128\3378209027.py:1: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df.dropna(inplace = True)
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.25)
# Splitting the data into training and testing data
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

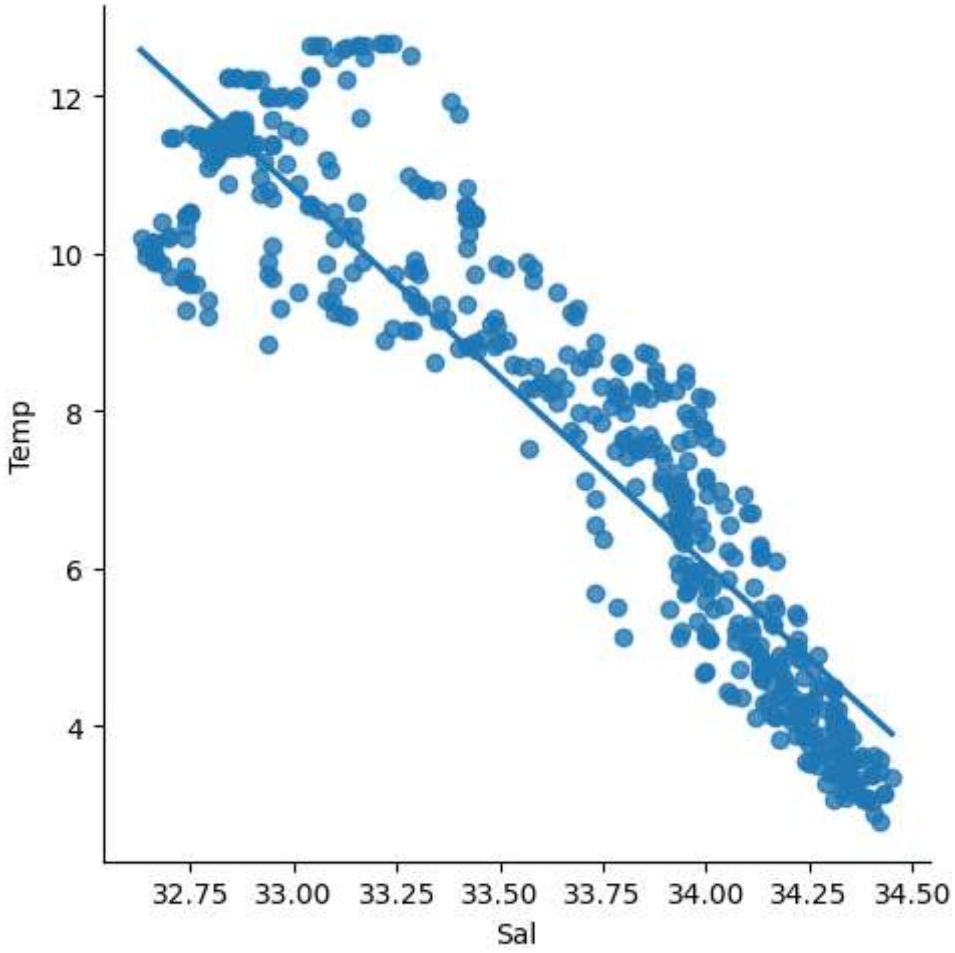
```
0.20471671221469934
```

In [11]:
```
# Step-6: Exploring Our results
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')
plt.show()
# Data scatter of predicted values
```



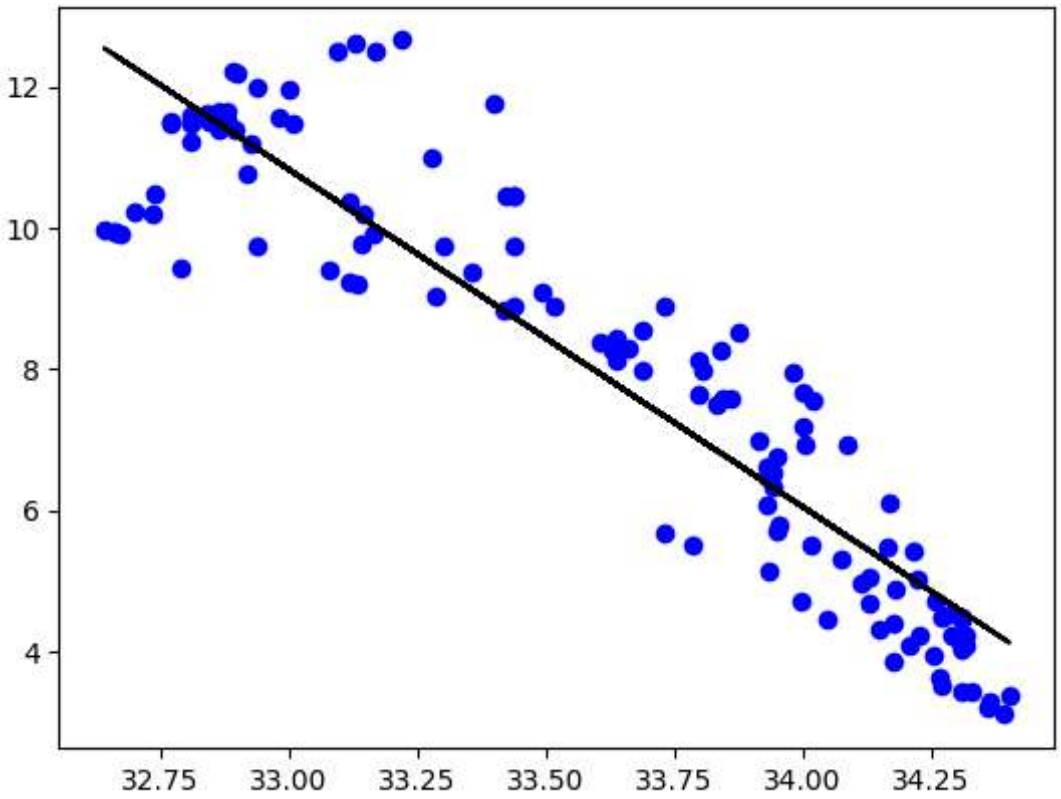
In [12]:
```
# step-7: Working with a smaller dataset
df500 = df[:][:500]
# selecting the 1st 500 rows of the data
sns.lmplot(x = "Sal", y ="Temp", data = df500, order = 1, ci = None)
```

Out[12]: <seaborn.axisgrid.FacetGrid at 0x1f73cab89a0>



In [13]:
```
df500.fillna(method ='ffill', inplace = True)
X = np.array(df500['Sal']).reshape(-1,1)
y = np.array(df500['Temp']).reshape(-1,1)
df500.dropna(inplace = True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
regr = LinearRegression()
regr.fit(X_train, y_train)
print("Regression: ",regr.score(X_test, y_test))
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color = 'k')
plt.show()
```

```
Regression:  0.8481877537701015
```



In [14]:
```
# Step 8: Evaluation of model

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score

# Train the model

model = LinearRegression()

model.fit(X_train, y_train)

y_pred=model.predict(X_test)

r2=r2_score(y_test,y_pred)

print("R2 score: ",r2)
# Evaluate the model on the test set
```

```
R2 score:  0.8481877537701015
```

Conclusion

Dataset we have taken is poor for linear model but with the smaller data works well with linear model.

flat vehicles dataset

In [15]:
```
# importing all the libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [16]:
```
df=pd.read_csv(r"C:\Users\91756\Documents\python\fiat500_VehicleSelection_Data
df
```

Out[16]:

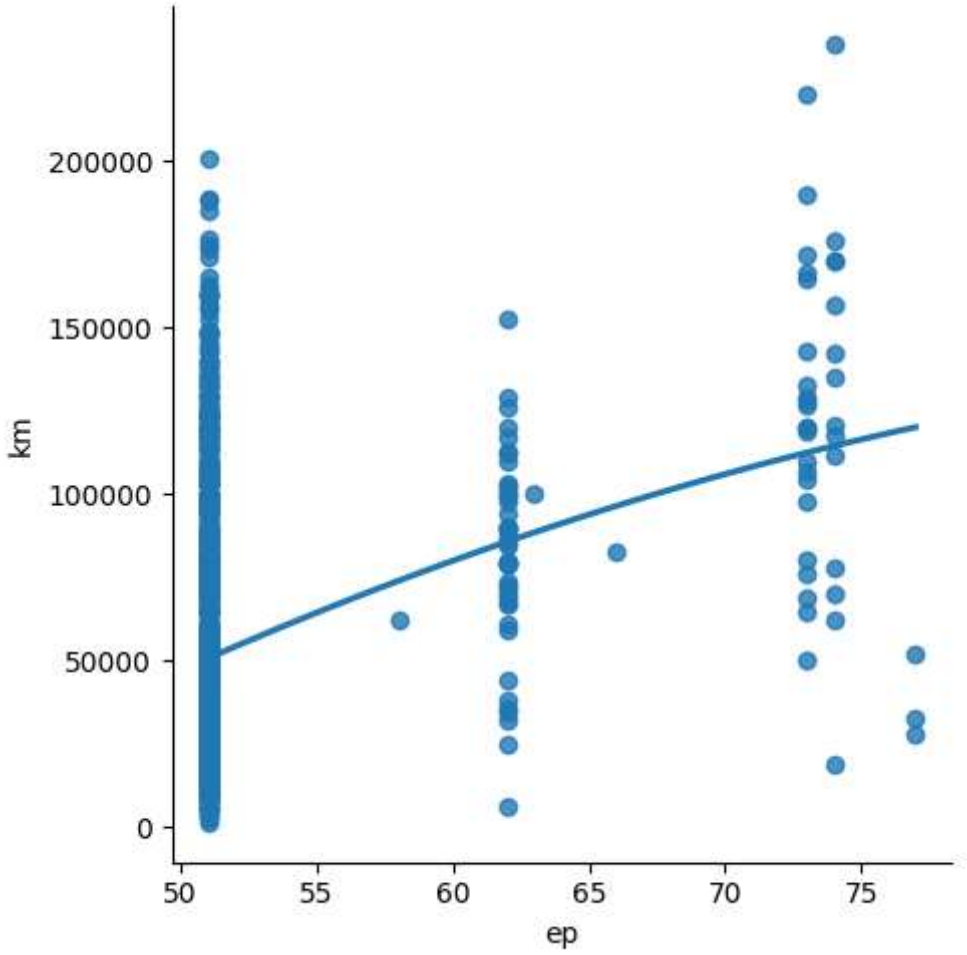
	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1	lounge	51	882	25000	1	44.907242	8.611560
1	2	pop	51	1186	32500	1	45.666359	12.241890
2	3	sport	74	4658	142228	1	45.503300	11.417840
3	4	lounge	51	2739	160000	1	40.633171	17.634609
4	5	pop	73	3074	106880	1	41.903221	12.495650
...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870
1535	1536	pop	51	2223	60457	1	45.481541	9.413480
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270
1537	1538	pop	51	1766	54276	1	40.323410	17.568270

1538 rows × 9 columns

In [17]:
```
df = df[['engine_power', 'km']]
df.columns=['ep','km']
```

In [18]:
```
sns.lmplot(x="ep", y="km", data=df, order=2, ci= None)
```

Out[18]: <seaborn.axisgrid.FacetGrid at 0x1f73f2d6050>



In [19]:
```
df.describe()
```

Out[19]:

	ep	km
count	1538.000000	1538.000000
mean	51.904421	53396.011704
std	3.988023	40046.830723
min	51.000000	1232.000000
25%	51.000000	20006.250000
50%	51.000000	39031.000000
75%	51.000000	79667.750000
max	77.000000	235000.000000

In [20]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ep      1538 non-null   int64
 1   km      1538 non-null   int64
dtypes: int64(2)
memory usage: 24.2 KB
```

In [21]:
```
df.fillna(method='ffill', inplace=True)
```

```
C:\Users\91756\AppData\Local\Temp\ipykernel_23128\3970806690.py:1: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df.fillna(method='ffill', inplace=True)
```

In [22]:
```
X = np.array(df['ep']).reshape(-1,1)
y = np.array(df['km']).reshape(-1,1)
```

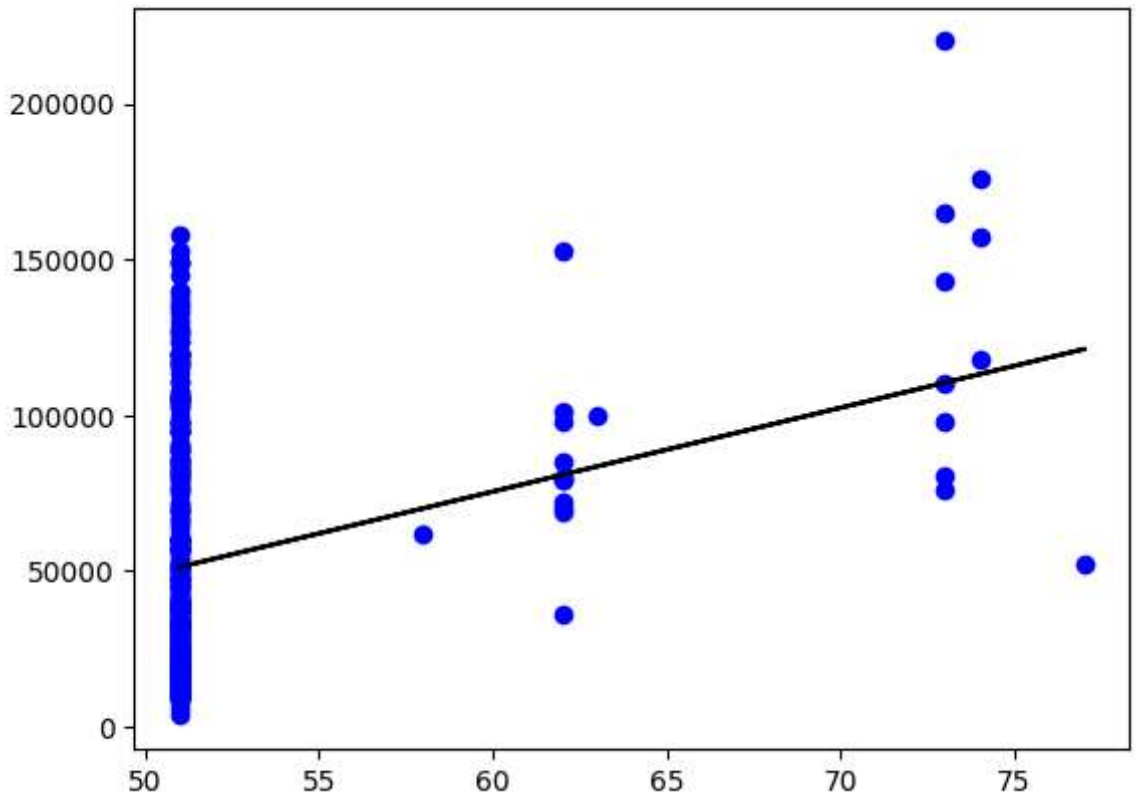
In [23]:
```
df.dropna(inplace = True)
```

```
C:\Users\91756\AppData\Local\Temp\ipykernel_23128\1791587065.py:1: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df.dropna(inplace = True)
```

In [24]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.25)
# Splitting the data into training and testing data
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

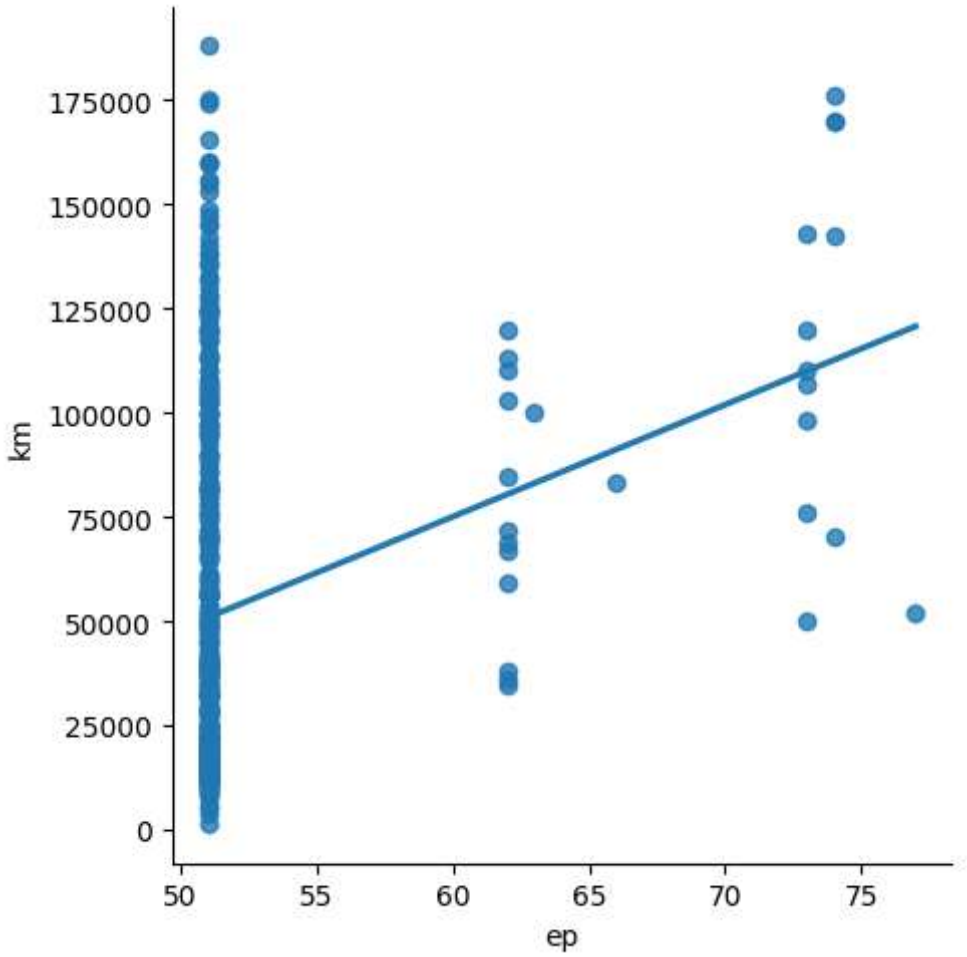
```
0.12086823703399341
```

In [25]:
```
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')
plt.show()
```



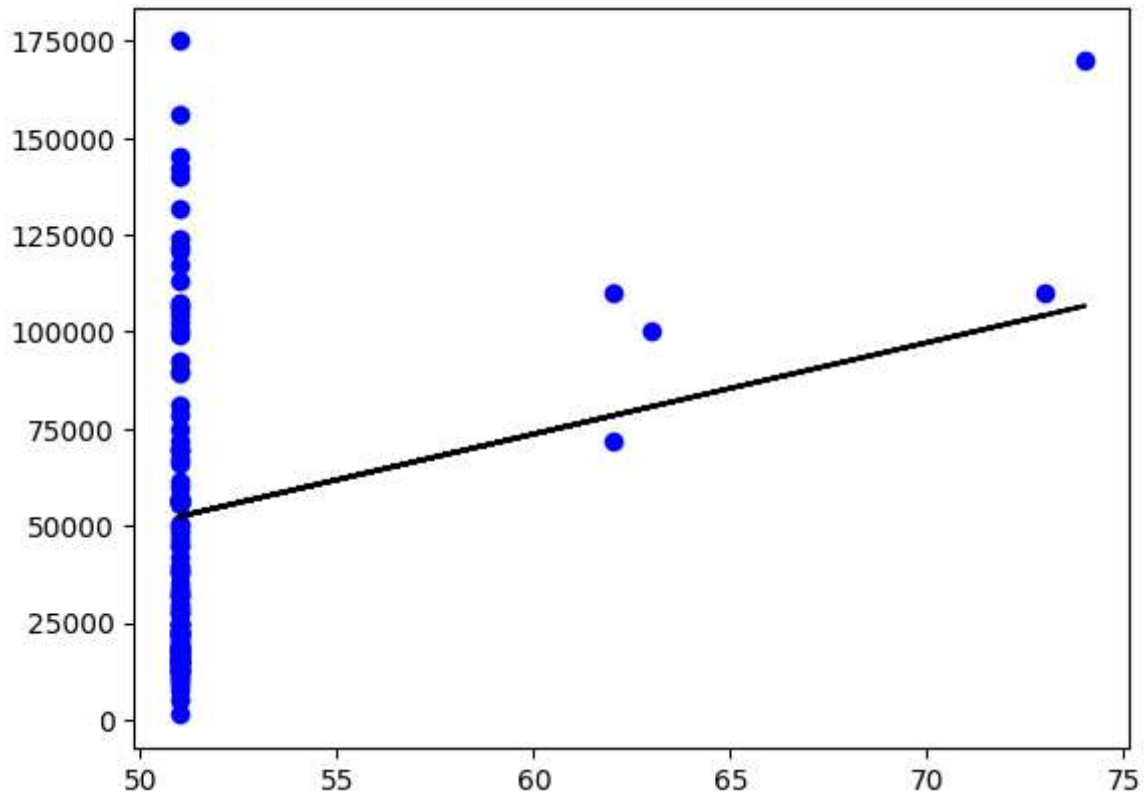
In [26]:
```
df500 = df[:][:500]
# selecting the 1st 500 rows of the data
sns.lmplot(x = "ep", y ="km", data = df500, order = 1, ci = None)
```

Out[26]: <seaborn.axisgrid.FacetGrid at 0x1f73f2d7430>



In [27]:
```
df500.fillna(method ='ffill', inplace = True)
X = np.array(df500['ep']).reshape(-1,1)
y = np.array(df500['km']).reshape(-1,1)
df500.dropna(inplace = True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
regr = LinearRegression()
regr.fit(X_train, y_train)
print("Regression: ",regr.score(X_test, y_test))
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color = 'k')
plt.show()
```

```
Regression:  0.077822590759271
```



In [28]:
```
# Step 8: Evaluation of model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
# Train the model
model = LinearRegression()
model.fit(X_train, y_train)
y_pred=model.predict(X_test)
r2=r2_score(y_test,y_pred)
print("R2 score: ",r2)
# Evaluate the model on the test set
```

```
R2 score:  0.077822590759271
```

Conclusion

Dataset we have taken is poor for linear model but with the smaller data works well with linear model.

USA Housing

Problem Statement

A realestate agents want help to predict the house price for regions in the USA. He gave you the data set to work on and you the Linear Regression model. Create a model that will help him to estimate of what the house would sell for

Data collection

The data set contains 7 columns and 5000 rows with .csv extension The data contains the following columns. 'Avg.Area Income'-Avg.The income of the house holder of the city house is located; 'Avg.Area House Age'-Avg.Age of house in the same city; 'Avg.Area Number Of Rooms'-Avg.Number of houses in the same city; 'Avg.Area Number Of Bedrooms'-Avg Number of bedrooms of houses in the same city; 'Area Population'-population of the city; 'Price'-Price of that the house sold at; 'Address'-Address of the houses;

In [29]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [30]:
```
df=pd.read_csv(r"C:\Users\91756\Downloads\USA_Housing.csv")
df
```

Out[30]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	A
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Fe 674\nLaurab
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnsor Suite 079 Kathleer
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Eli Stravenue\nDani WI 0
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nF
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymonc AE
...	...	...	...	...	...	...	
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams AP 3015
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 925 8489\nAPO AA
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy (Suite 076\nJoshu \
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nF
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Apt. 509\nEas

5000 rows × 7 columns

In [31]:
```
df.head()
```

Out[31]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Addre
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry A 674\nLaurabury, 370
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Vie Suite 079\nLa Kathleen, C
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizab Stravenue\nDanielto WI 0648
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO 448
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nF AE 093

In [32]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Avg. Area Income             5000 non-null   float64
 1   Avg. Area House Age          5000 non-null   float64
 2   Avg. Area Number of Rooms    5000 non-null   float64
 3   Avg. Area Number of Bedrooms 5000 non-null   float64
 4   Area Population              5000 non-null   float64
 5   Price                        5000 non-null   float64
 6   Address                      5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

In [33]:
```
df.describe()
```

Out[33]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

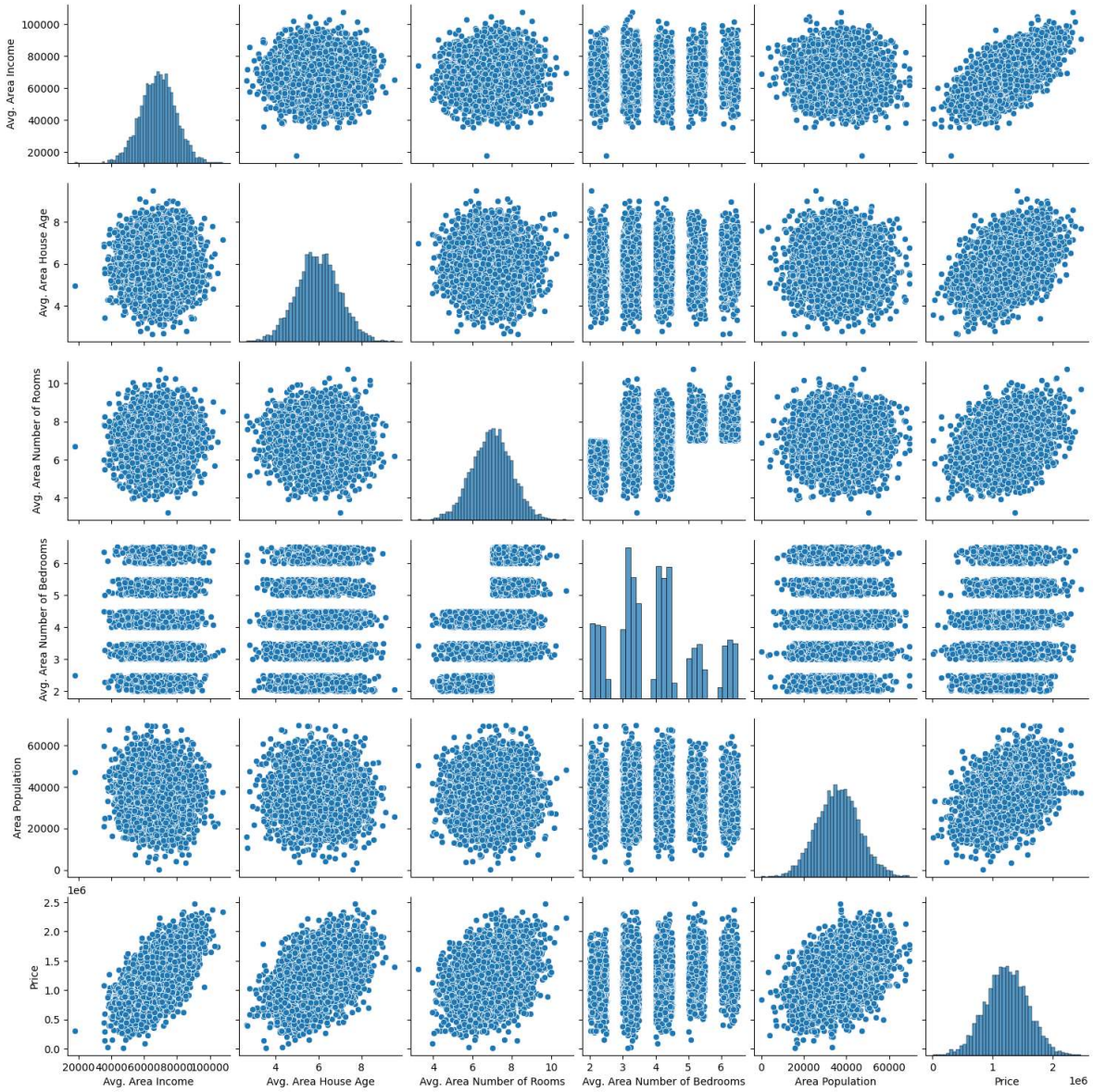
In [34]:
```
df.columns
```

Out[34]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Room
s',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Addres
s'],
      dtype='object')
```

Exploratory Data Analysis

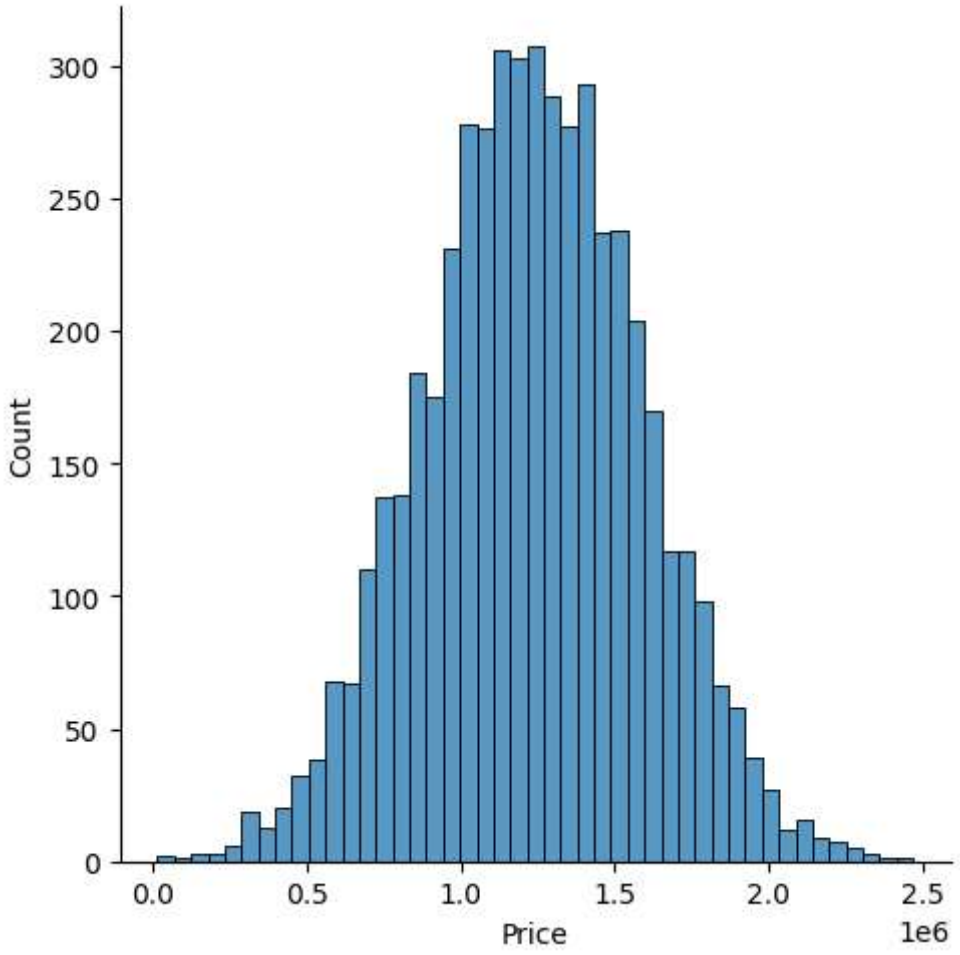
In [35]:
```
sns.pairplot(df)
```

Out[35]: <seaborn.axisgrid.PairGrid at 0x1f73cc6fe80>



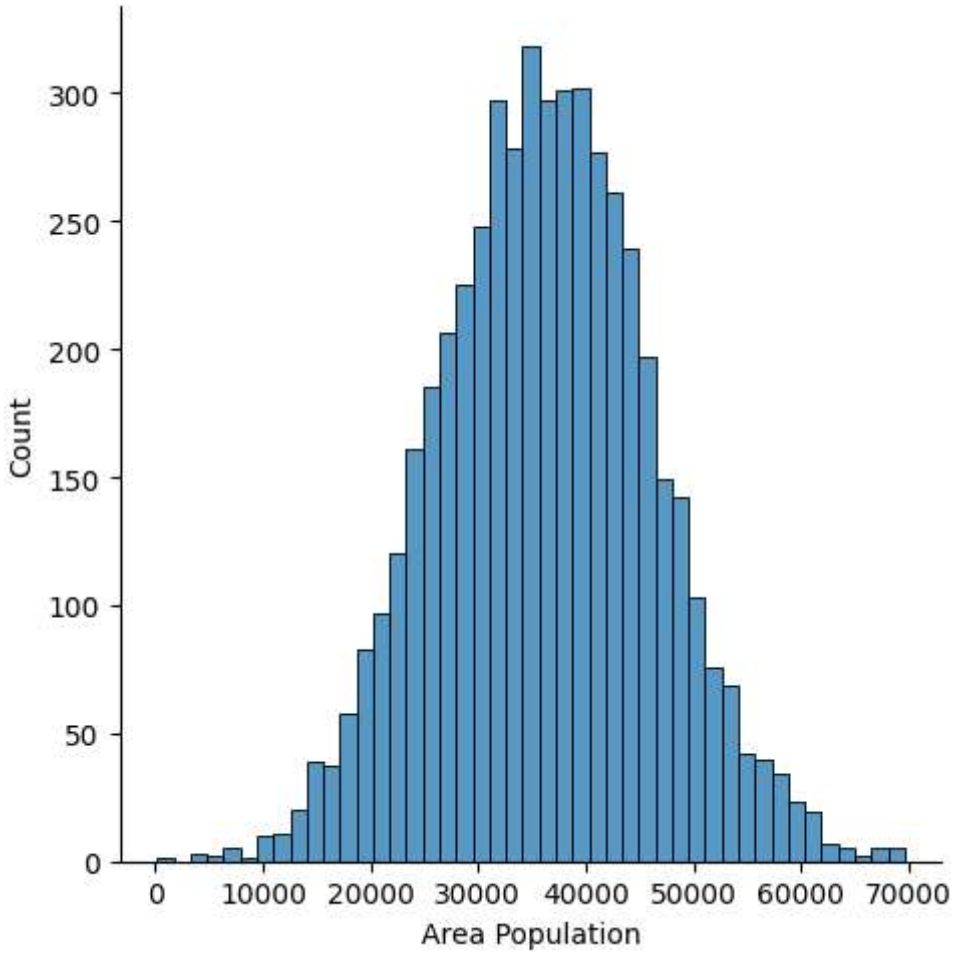
In [36]:
```
sns.displot(df['Price'])
```

Out[36]: <seaborn.axisgrid.FacetGrid at 0x1f7645ab550>



In [37]:
```
sns.displot(df['Area Population'])
```

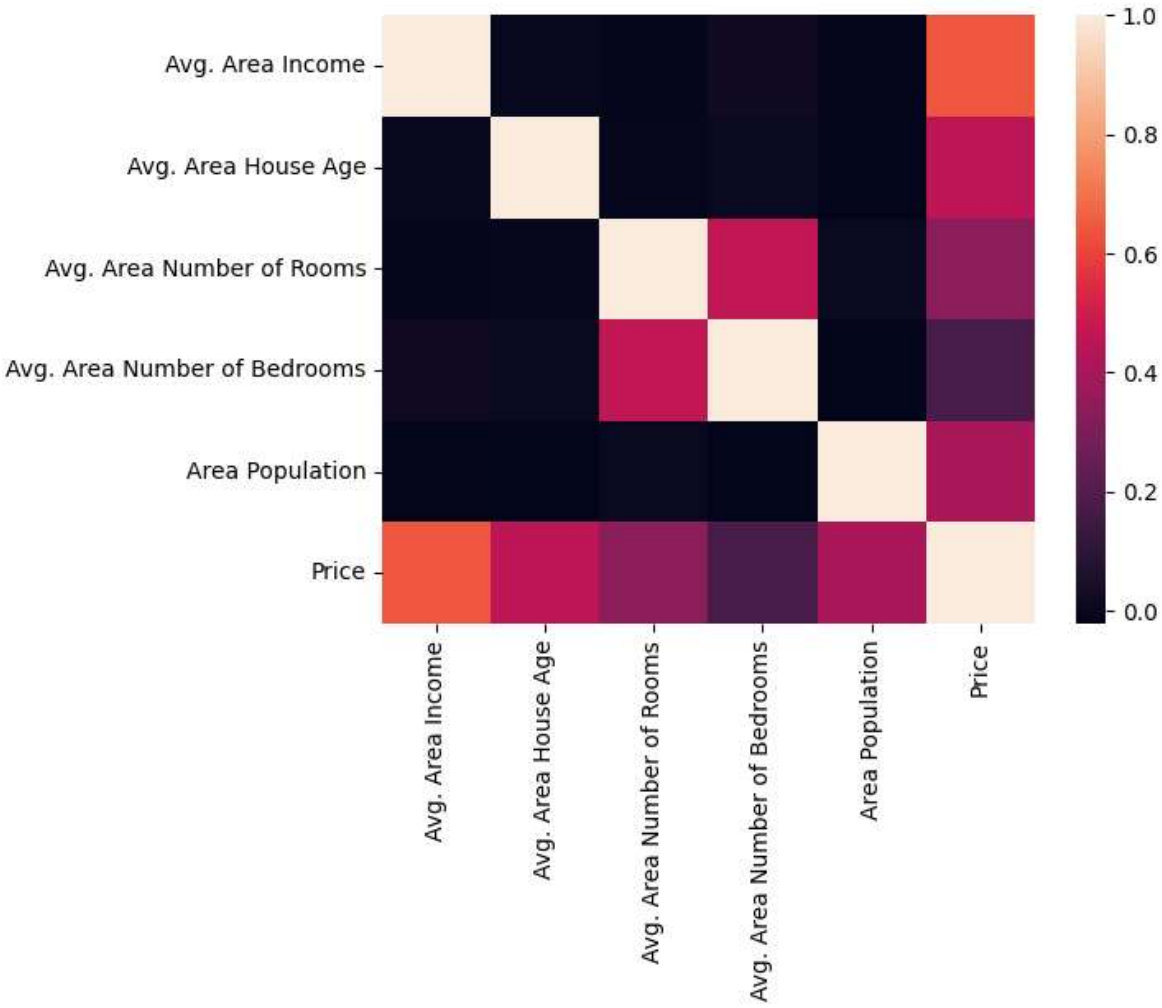
Out[37]: <seaborn.axisgrid.FacetGrid at 0x1f73cc551e0>



In [38]:
```
Housedf=df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Ro
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price']]
```

In [39]:
```
sns.heatmap(Housedf.corr())
```

Out[39]: <Axes: >



To train the Model

We are going to train linear regression model.We need to first split up our data into X list that contains the features to train on, and a Y list with the target variable in this case, the price column.We will ignore the address column because it only has text which is not useful for Linear Regression model

In [40]:
```
x=Housedf[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Roo
       'Avg. Area Number of Bedrooms', 'Area Population']]
y=df['Price']
```

In [41]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=
```

In [42]:
```
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(x_train,y_train)
```

Out[42]:
```
LinearRegression
LinearRegression()
```

In [43]:
```
print(lm.intercept_)
```

```
-2641372.6673013885
```

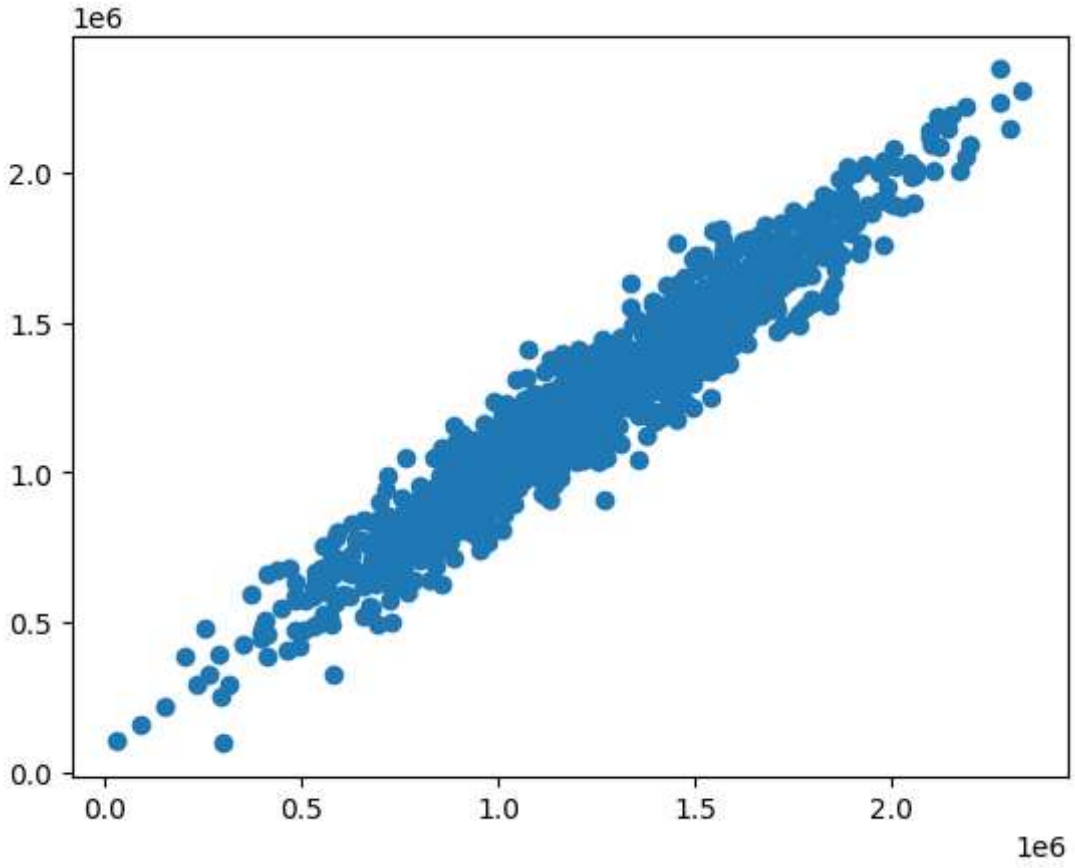
In [44]:
```
coeff_df=pd.DataFrame(lm.coef_,x.columns,columns=['coefficient'])
coeff_df
```

Out[44]:

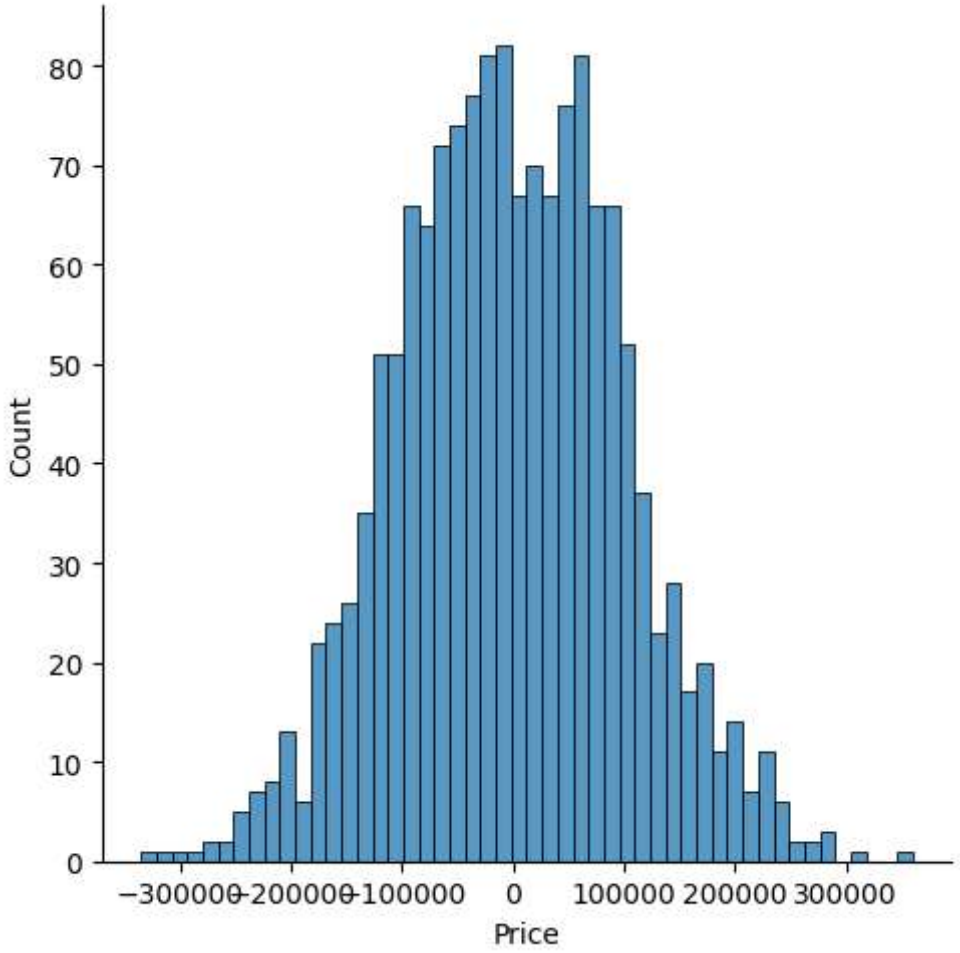
	coefficient
Avg. Area Income	21.617635
Avg. Area House Age	165221.119872
Avg. Area Number of Rooms	121405.376596
Avg. Area Number of Bedrooms	1318.718783
Area Population	15.225196

In [45]:
```
predictions=lm.predict(x_test)
plt.scatter(y_test,predictions)
```

Out[45]:
```
<matplotlib.collections.PathCollection at 0x1f7651bd3f0>
```



In [46]:
```
sns.displot((y_test-predictions),bins=50);
```



In [47]:
```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,predictions))
print('MSE:',metrics.mean_squared_error(y_test,predictions))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
MAE: 81257.55795855928
MSE: 10169125565.897568
RMSE: 100842.0823163503
```

logistics Regression

In [48]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [49]:
```
df=pd.read_csv(r"C:\Users\91756\Downloads\ionosphere_data (1).csv")
df
```

Out[49]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	co
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0
...	...	...	...	...	...	...	...	...	
346	True	False	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0
347	True	False	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0
348	True	False	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0
349	True	False	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0
350	True	False	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0

351 rows × 35 columns

In [50]:
```
pd.set_option('display.max_rows',100000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [51]:
```
print('This DataFrame has %d Rows and %d columns'%(df.shape))
```

```
This DataFrame has 351 Rows and 35 columns
```

In [52]:
```
df.head()
```

Out[52]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	colur
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.0(
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.0(
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.8
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.0(
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.7

In [53]:
```
features_matrix = df.iloc[:,0:34]
```

In [54]:
```
target_vector = df.iloc[:,-1]
```

In [55]:
```
print('The Features Matrix Has %d Rows And %d Column(S)'%(features_matrix.shap
print('The Target Matrix Has %d Rows and %d Column(s)'%(np.array(target_vector
```

```
The Features Matrix Has 351 Rows And 34 Column(S)
The Target Matrix Has 351 Rows and 1 Column(s)
```

In [56]:
```
features_matrix_standardized = StandardScaler().fit_transform(features_matrix)
```

In [57]:
```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_interc
```

In [58]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_ve
```

In [59]:
```
observation=[[1,0,0.99539,-0.05889,0.8524299999999999,0.02306,0.83397999999999
 -0.37708,1.0,0.0376,0.8524299999999999,-0.17755,0.59755,-0.44945,
 0.60536,-0.38223,0.8435600000000001,-0.38542,0.58212,-0.32192,0.56971
 ,-0.29674,0.36946,-0.47357,0.56811,-0.51171,0.41078000000000003,
 -0.46168000000000003,0.21266,-0.3409,0.42267,-0.54487,0.18641,
 -0.453]]
```

In [60]:
```
predictions = Logistic_Regression_Model.predict(observation)
print('The Model predicted The observation To Belong To class %s'%(predictions
```

```
The Model predicted The observation To Belong To class ['g']
```

In [61]:
```
print('The Algorithm Was Trained To predict one of the two classes:%s'%(algori
```

```
The Algorithm Was Trained To predict one of the two classes:['b' 'g']
```

In [62]:
```
print("""The Model says The probability of the observation We passed Belonging
print()
print("""The Model says The probability of the observation We passed Belonging
```

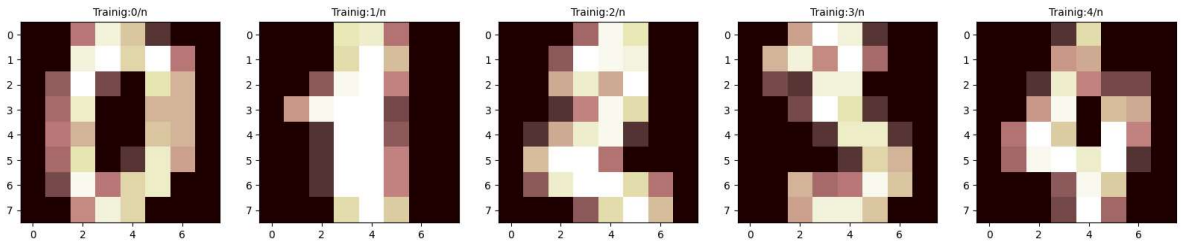
```
The Model says The probability of the observation We passed Belonging To cla
ss['g'] Is 0.00777393160013784

The Model says The probability of the observation We passed Belonging To cla
ss['b'] Is 0.00777393160013784
```

In [63]:
```
import re
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
%matplotlib inline
digits=load_digits()
```

In [64]:
```
print("Image Data Shape",digits.data.shape)
print("Label Data Shape",digits.target.shape)
```

```
Image Data Shape (1797, 64)
Label Data Shape (1797,)
```

In [65]:
```
plt.figure(figsize=(20,4))
for index,(image,label) in enumerate(zip(digits.data[0:5],digits.target[0:5]))
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.pink)
    plt.title('Trainig:%i/n'%label,fontsize=10)
```



In [66]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_
```

In [67]:
```
print(x_train.shape)
```

```
(1257, 64)
```

In [68]:
```
print(y_train.shape)
```

```
(1257,)
```

In [69]:
```
print(x_test.shape)
```

```
(540, 64)
```

In [70]:
```
print(y_test.shape)
```

```
(540,)
```

In [71]:
```
from sklearn.linear_model import LogisticRegression
```

In [72]:
```
LogisticRegr=LogisticRegression(max_iter=10000)
LogisticRegr.fit(x_train,y_train)
```

Out[72]:
```
LogisticRegression
LogisticRegression(max_iter=10000)
```

In [73]:
```
print(LogisticRegr.predict(x_test))
```

```
[4 0 9 1 8 7 1 5 1 6 6 7 6 1 5 5 8 6 2 7 4 6 4 1 5 2 9 5 4 6 5 6 3 4 0 9 9
 8 4 6 8 8 5 7 9 8 9 6 1 7 0 1 9 7 3 3 1 8 8 8 9 8 5 8 4 9 3 5 8 4 3 1 3 8
 7 3 3 0 8 7 2 8 5 3 8 7 6 4 6 2 2 0 1 1 5 3 5 7 1 8 2 2 6 4 6 7 3 7 3 9 4
 7 0 3 5 1 5 0 3 9 2 7 3 2 0 8 1 9 2 1 5 1 0 3 4 3 0 8 3 2 2 7 3 1 6 7 2 8
 3 1 1 6 4 8 2 1 8 4 1 3 1 1 9 5 4 8 7 4 8 9 5 7 6 9 4 0 4 0 0 9 0 6 5 8 8
 3 7 9 2 0 8 2 7 3 0 2 1 9 2 7 0 6 9 3 1 1 3 5 2 5 5 2 1 2 9 4 6 5 5 5 9 7
 1 5 9 6 3 7 1 7 5 1 7 2 7 5 5 4 8 6 6 2 8 7 3 7 8 0 9 5 7 4 3 4 1 0 3 3 5
 4 1 3 1 2 5 1 4 0 3 1 5 5 7 4 0 1 0 9 5 5 5 4 0 1 8 6 2 1 1 1 7 9 6 7 9 7
 0 4 9 6 9 2 7 2 1 0 8 2 8 6 5 7 8 4 5 7 8 6 4 2 6 9 3 0 0 8 0 6 6 7 1 4 5
 6 9 7 2 8 5 1 2 4 1 8 8 7 6 0 8 0 6 1 5 7 8 0 4 1 4 5 9 2 2 3 9 1 3 9 3 2
 8 0 6 5 6 2 5 2 3 2 6 1 0 7 6 0 6 2 7 0 3 2 4 2 3 6 9 7 7 0 3 5 4 1 2 2 1
 2 7 7 0 4 9 8 5 6 1 6 5 2 0 8 2 4 3 3 2 9 3 8 9 9 5 9 0 3 4 7 9 8 5 7 5 0
 5 3 5 0 2 7 3 0 4 3 6 6 1 9 6 3 4 6 4 6 7 2 7 6 3 0 3 0 1 3 6 1 0 4 3 8 4
 3 3 4 8 6 9 6 3 3 0 5 7 8 9 1 5 3 2 5 1 7 6 0 6 9 5 2 4 4 7 2 0 5 6 2 0 8
 4 4 4 7 1 0 4 1 9 2 1 3 0 5 3 9 8 2 6 0 0 4]
```

In [74]:
```
score=LogisticRegr.score(x_test,y_test)
print(score)
```

```
0.9537037037037037
```

In [75]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [76]:
```
df=pd.read_csv(r"C:\Users\91756\Documents\python\gender_submission.csv")
df
```

Out[76]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1
5	897	0
6	898	1
7	899	0
8	900	1
9	901	0
10	902	0

In [77]:
```
pd.set_option('display.max_rows',100000000000)
pd.set_option('display.max_columns',100000000000)
pd.set_option('display.width',95)
```

In [78]:
```
print('This DataFrame has %d Rows and %d columns'%(df.shape))
```

```
This DataFrame has 418 Rows and 2 columns
```

In [79]:
```
df.head()
```

Out[79]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1

In [80]:
```
features_matrix=df.iloc[:,0:34]
```

In [81]:
```
target_vector=df.iloc[:,-1]
```

In [82]:
```
print('The features Matrix Has %d Rows and %d Columns(s)'%(features_matrix.sha
print('The Target Matrix has %d Rows and %d Columns(s)'%(np.array(target_vecto
```

```
The features Matrix Has 418 Rows and 2 Columns(s)
The Target Matrix has 418 Rows and 1 Columns(s)
```

In [83]:
```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [84]:
```
algorithm=LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_interc
```

In [85]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_ve
```

In [86]:
```
observation=[[0.4,1]]
```

In [87]:
```
predictions = Logistic_Regression_Model.predict(observation)
print('The Model predicted The observation To Belong To class %s'%(predictions
```

```
The Model predicted The observation To Belong To class [1]
```

In [88]:
```
print('The Algorithm Was Trained To predict one of the two classes:%s'%(algori
```

```
The Algorithm Was Trained To predict one of the two classes:[0 1]
```

In [89]:
```
print("""The Model says The probability of the observation We passed Belonging
print()
print("""The Model says The probability of the observation We passed Belonging
```

```
The Model says The probability of the observation We passed Belonging To cla
ss['0'] Is 0.0549611529834666

The Model says The probability of the observation We passed Belonging To cla
ss['1'] Is 0.0549611529834666
```

Data Analysis

To Predict and Analyse which gender has a High change of survival at the time of disaster.

Import datasets python packages and libraries

In [2]:
```
import numpy as np
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
# plt.rc("font",size=14)
import seaborn as sns
sns.set(style="white") # white background style for seaborn plots.
sns.set(style="whitegrid",color_codes=True)
import warnings
warnings.simplefilter(action='ignore')
```

In [3]:
```
train_df=pd.read_csv(r"C:\Users\91756\Documents\python\train.gender_submissior
train_df
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500
...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500

891 rows × 12 columns

In [7]:
```
test_df=pd.read_csv(r"C:\Users\91756\Documents\python\test.gender_submission.c
test_df
```

Out[7]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN
...	...	...	...	...	...	...	...	...	...	...
413	1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN
414	1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN
416	1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN
417	1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN

418 rows × 11 columns

In [93]:
```
train_df.head()
```

Out[93]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [94]:
```
train_df.shape
```

Out[94]: (891, 12)

In [95]:
```
test_df.head()
```

Out[95]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embar
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	

In [96]:
```
test_df.shape
```

Out[96]:
```
(418, 11)
```

In [8]:
```
train_df.describe
```

Out[8]:
```
<bound method NDFrame.describe of      PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp
0                              Braund, Mr. Owen Harris    male  22.0      1
\
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 ...     ...   ...    ...
886                             Montvila, Rev. Juozas    male  27.0      0
887                      Graham, Miss. Margaret Edith  female  19.0      0
888          Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                             Behr, Mr. Karl Howell    male  26.0      0
890                               Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     ...               ...      ...   ...      ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]>
```

In [98]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

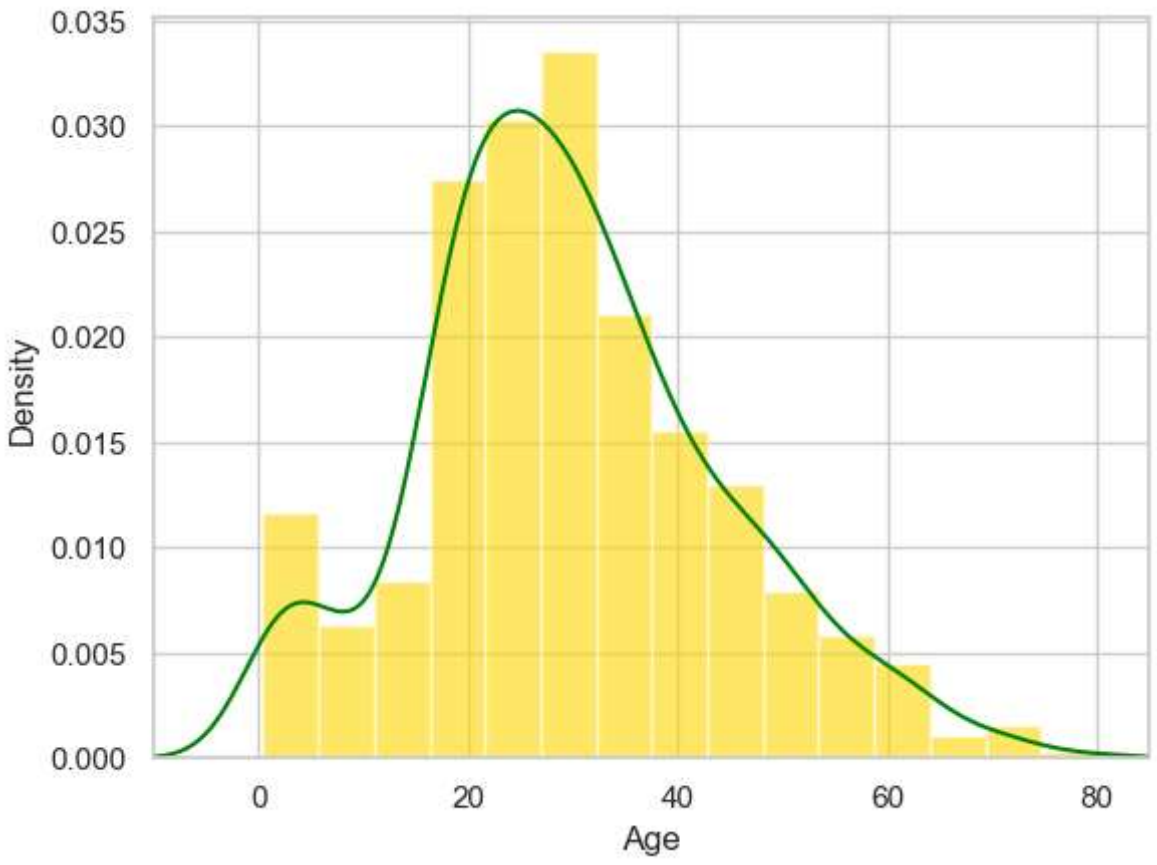
In [99]:
```
test_df.describe
```

Out[99]:
```
<bound method NDFrame.describe of      PassengerId  Pclass                                          
Name     Sex    Age
0            892       3                                  Kelly, Mr. Jam
es    male  34.50  \
1            893       3                  Wilkes, Mrs. James (Ellen Need
s)  female  47.00
2            894       2                        Myles, Mr. Thomas Franc
is    male  62.00
3            895       3                                  Wirz, Mr. Albe
rt    male  27.00
4            896       3       Hirvonen, Mrs. Alexander (Helga E Lindqvis
t)  female  22.00
5            897       3                       Svensson, Mr. Johan Cerv
in    male  14.00
6            898       3                             Connolly, Miss. Ka
te  female  30.00
7            899       2                     Caldwell, Mr. Albert Franc
is    male  26.00
8            900       3          Abrahim, Mrs. Joseph (Sophie Halaut Eas
```

In [100]:
```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
 10  Embarked     418 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

In [9]:
```
ax=train_df["Age"].hist(bins=15,density=True,stacked=True,color='gold',alpha=0
train_df["Age"].plot(kind='density',color='green')
ax.set(xlabel='Age')
plt.xlim(-10,85)
plt.show()
```



In [10]:
```
print(train_df["Age"].mean(skipna=True))
print(train_df["Age"].median(skipna=True))
```

```
29.69911764705882
28.0
```

In [11]:
```
print((train_df['Cabin'].isnull().sum()/train_df.shape[0])*100)
```

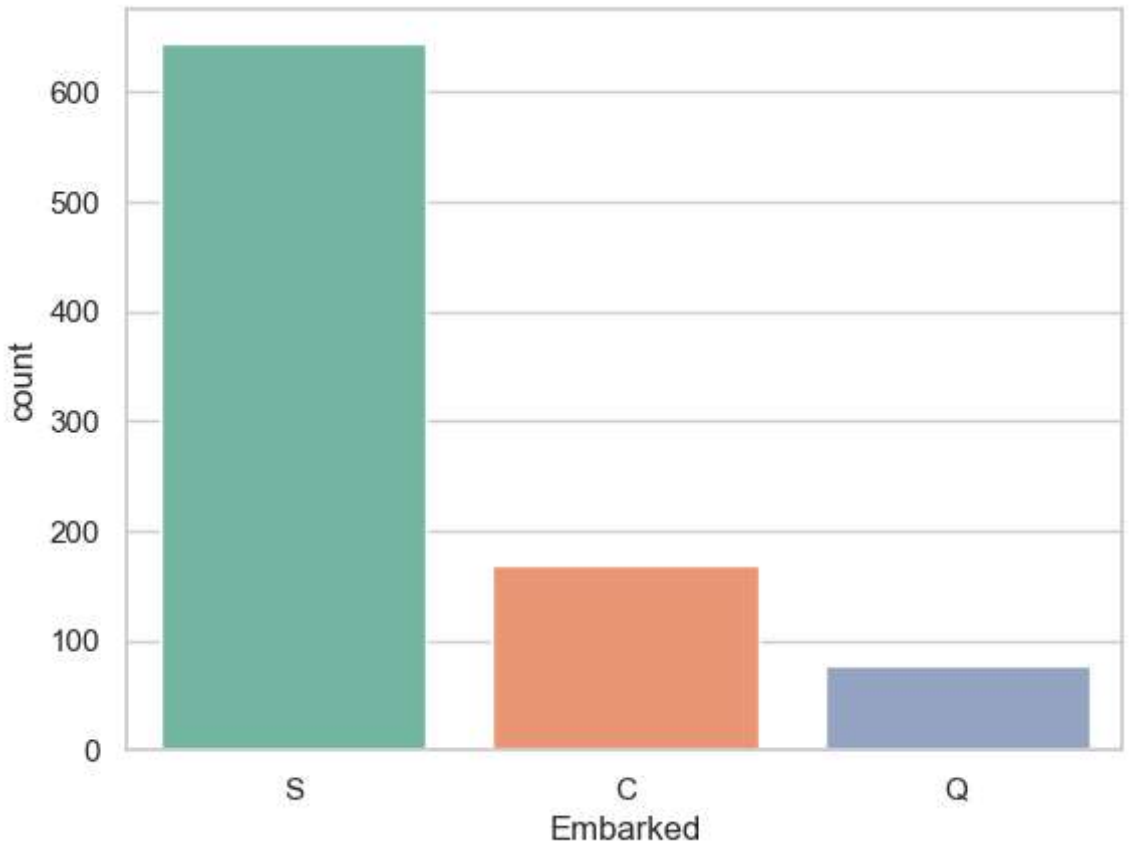
```
77.10437710437711
```

In [12]:
```
print((train_df['Embarked'].isnull().sum()/train_df.shape[0])*100)
```

```
0.22446689113355783
```

In [13]:
```
print('Boarded passengers grouped by port of embarkation(c=Cherbourg,Q=Queenst
print(train_df['Embarked'].value_counts())
sns.countplot(x='Embarked',data=train_df,palette='Set2')
plt.show()
```

```
Boarded passengers grouped by port of embarkation(c=Cherbourg,Q=Queenstown,s
=Southampton):
Embarked
S    644
C    168
Q     77
Name: count, dtype: int64
```



In [14]:
```
print(train_df['Embarked'].value_counts().idxmax())
```

```
S
```

In [15]:
```
train_data = train_df.copy()
```

In [16]:
```
train_data['Age'].fillna(train_df["Age"].median(skipna=True),inplace=True)
train_data["Embarked"].fillna(train_df['Embarked'].value_counts().idxmax(),inp
train_data.drop('Cabin',axis=1,inplace=True)
```

In [17]:
```
train_data.isnull().sum()
```

Out[17]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

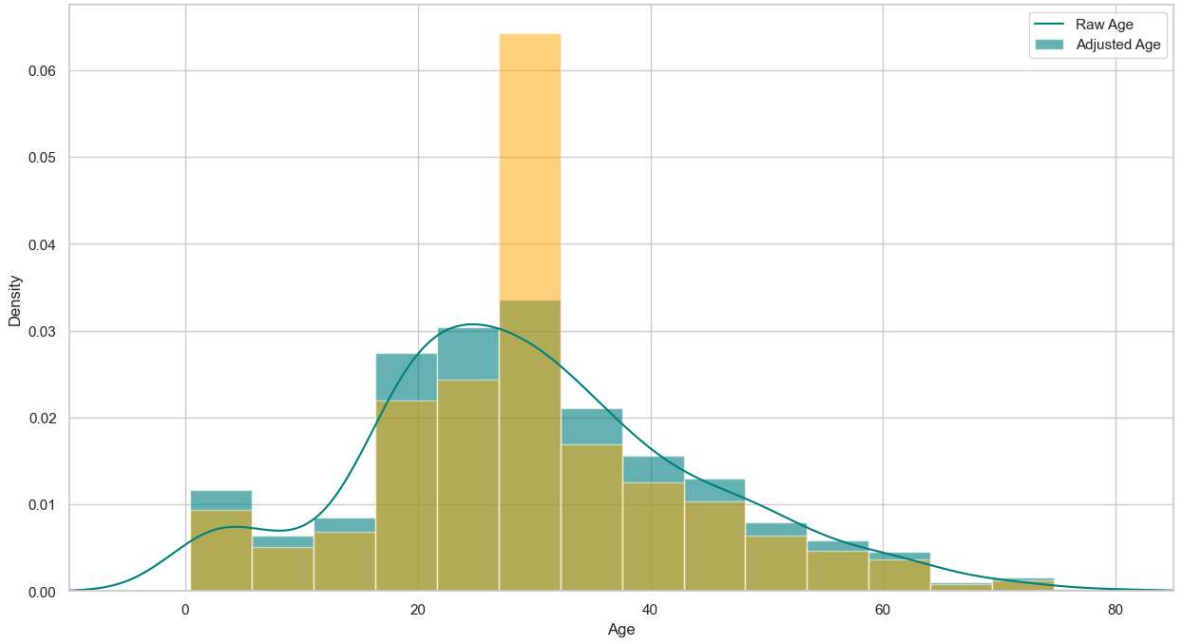
In [110]:
```
train_data.head()
```

Out[110]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ei
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [111]:
```
plt.figure(figsize= (15,8))
ax = train_df["Age"].hist(bins=15,density=True,stacked=True,color='teal',alpha
train_df["Age"].plot(kind='density',color='teal')
ax =train_data["Age"].hist(bins=15,density=True,stacked=True,color='orange',al
ax.legend(['Raw Age','Adjusted Age'])
ax.set(xlabel='Age')
plt.xlim(-10,85)
```

Out[111]: (-10.0, 85.0)



In [112]:
```
train_data['TravelAlone']=np.where((train_data["SibSp"]+train_data["Parch"])>0
train_data.drop('SibSp',axis=1,inplace=True)
train_data.drop('Parch',axis=1,inplace=True)
```

In [113]:
```
training=pd.get_dummies(train_data,columns=["Pclass","Embarked","Sex"])
training.drop('Sex_female',axis=1,inplace=True)
training.drop('PassengerId',axis=1,inplace=True)
training.drop('Name',axis=1,inplace=True)
training.drop('Ticket',axis=1,inplace=True)
final_train=training
final_train.head()
```

Out[113]:

	Survived	Age	Fare	TravelAlone	Pclass_1	Pclass_2	Pclass_3	Embarked_C	Embarked_
0	0	22.0	7.2500	0	False	False	True	False	Fals
1	1	38.0	71.2833	0	True	False	False	True	Fals
2	1	26.0	7.9250	1	False	False	True	False	Fals
3	1	35.0	53.1000	0	True	False	False	False	Fals
4	0	35.0	8.0500	1	False	False	True	False	Fals

In [114]:
```
test_df.isnull().sum()
```

Out[114]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

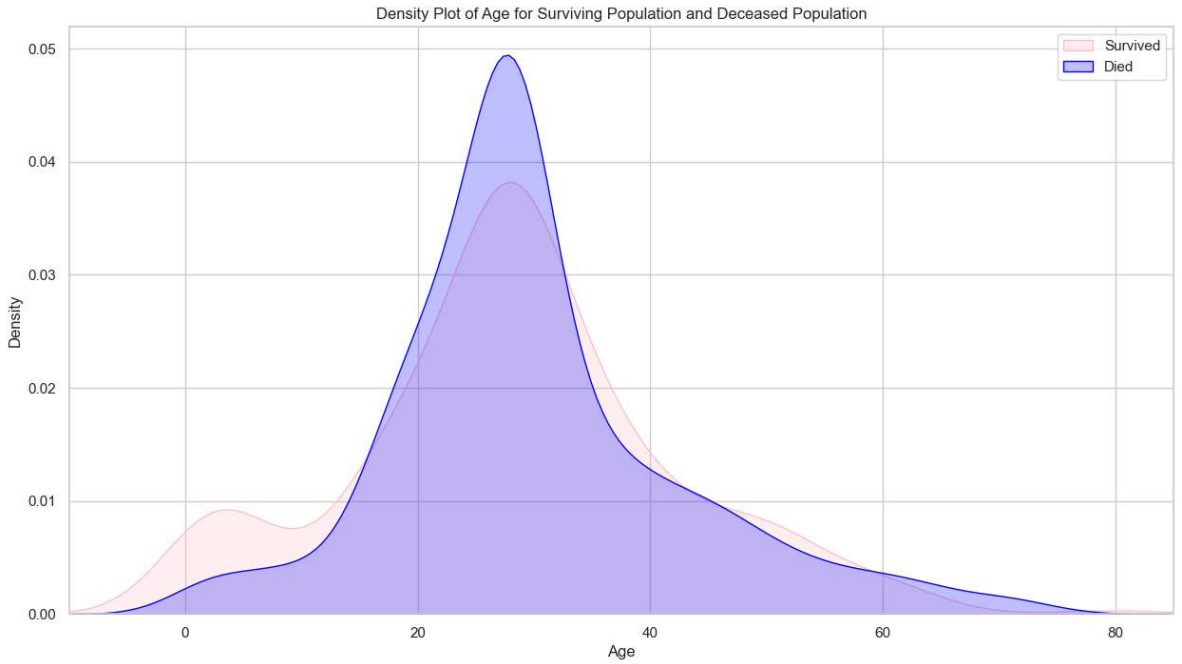
In [115]:
```
test_data = test_df.copy()
test_data["Age"].fillna(train_df["Age"].median(skipna=True), inplace=True)
test_data["Fare"].fillna(train_df["Fare"].median(skipna=True), inplace=True)
test_data.drop('Cabin', axis=1, inplace=True)
test_data['TravelAlone']=np.where((test_data["SibSp"]+test_data["Parch"])>0, 0
test_data.drop('SibSp', axis=1, inplace=True)
test_data.drop('Parch', axis=1, inplace=True)
testing = pd.get_dummies(test_data, columns=["Pclass","Embarked","Sex"])
testing.drop('Sex_female', axis=1, inplace=True)
testing.drop('PassengerId', axis=1, inplace=True)
testing.drop('Name', axis=1, inplace=True)
testing.drop('Ticket', axis=1, inplace=True)
final_test = testing
final_test.head()
```

Out[115]:

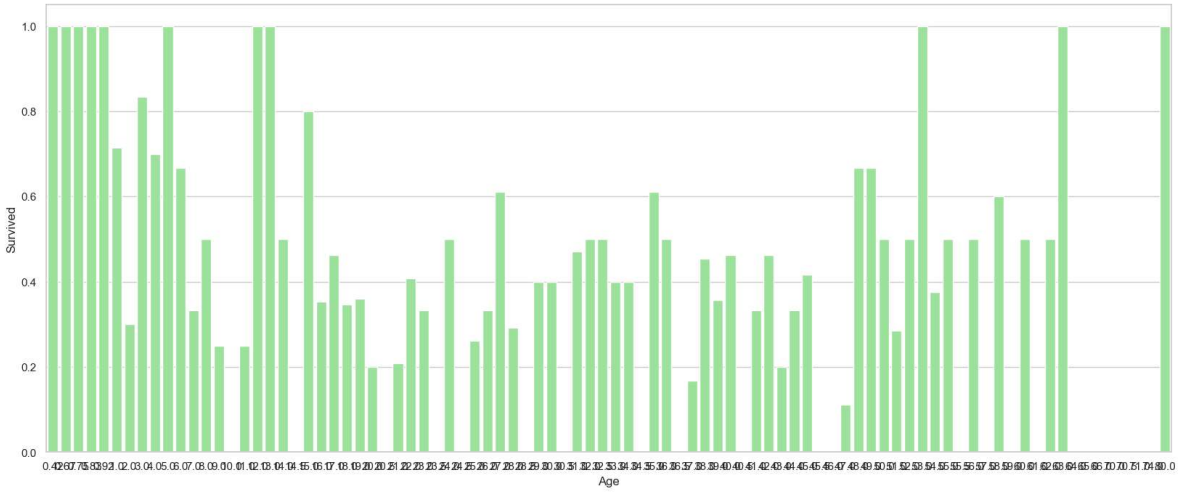
	Age	Fare	TravelAlone	Pclass_1	Pclass_2	Pclass_3	Embarked_C	Embarked_Q	Embarl
0	34.5	7.8292	1	False	False	True	False	True	
1	47.0	7.0000	0	False	False	True	False	False	
2	62.0	9.6875	1	False	True	False	False	True	
3	27.0	8.6625	1	False	False	True	False	False	
4	22.0	12.2875	0	False	False	True	False	False	

EXPLORATORY DATA ANALYSIS

In [116]:
```
plt.figure(figsize=(15,8))
ax=sns.kdeplot(final_train["Age"][final_train.Survived ==1],color="pink",shade
sns.kdeplot(final_train["Age"][final_train.Survived == 0], color="blue", shade
plt.legend(['Survived', 'Died'])
plt.title('Density Plot of Age for Surviving Population and Deceased Populatio
ax.set(xlabel='Age')
plt.xlim(-10,85)
plt.show()
```



In [117]:
```
plt.figure(figsize=(20,8))
avg_survival_byage=final_train[["Age", "Survived"]].groupby(['Age'], as_index=
g=sns.barplot(x='Age', y='Survived', data=avg_survival_byage, color="LightGree
plt.show()
```



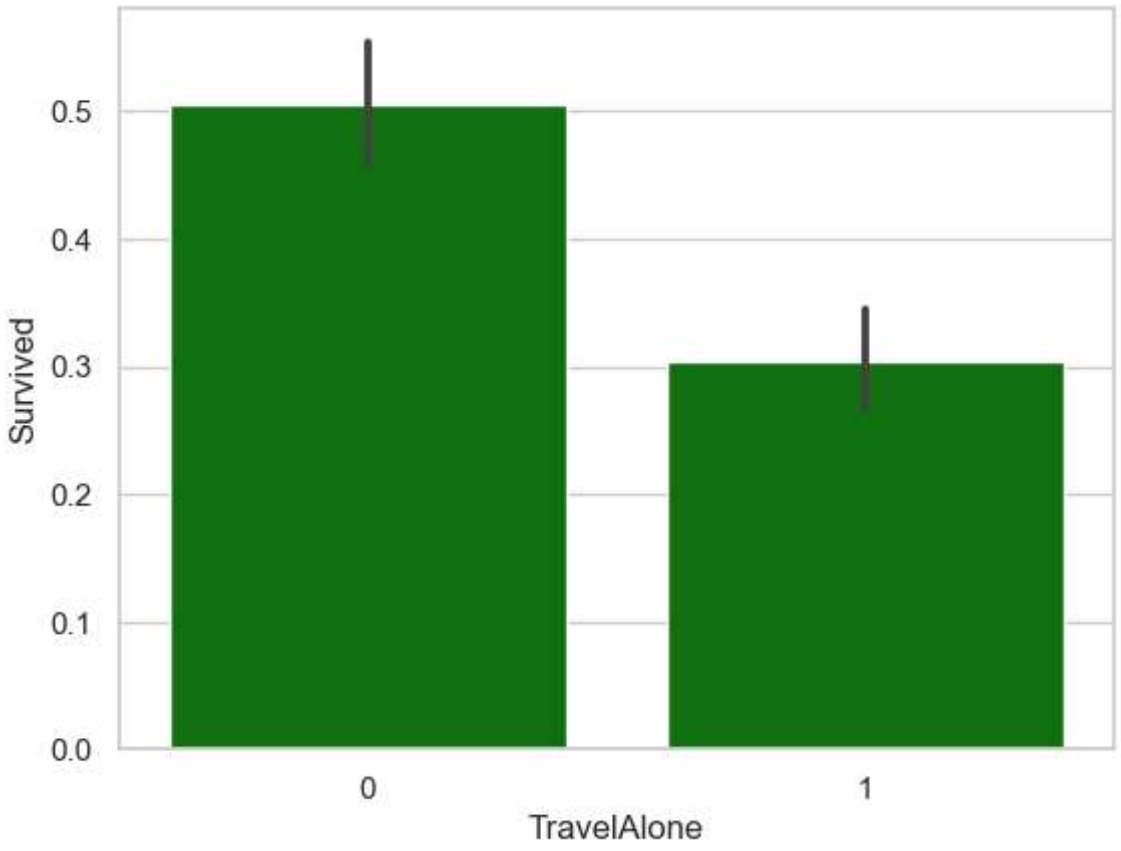
In [118]:
```
final_train['IsMinor']=np.where(final_train['Age']<=16, 1, 0)
print(final_train['IsMinor'])
```

```
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     1
8     0
9     1
10    1
11    0
12    0
13    0
14    1
15    0
16    1
17    0
18    0
```

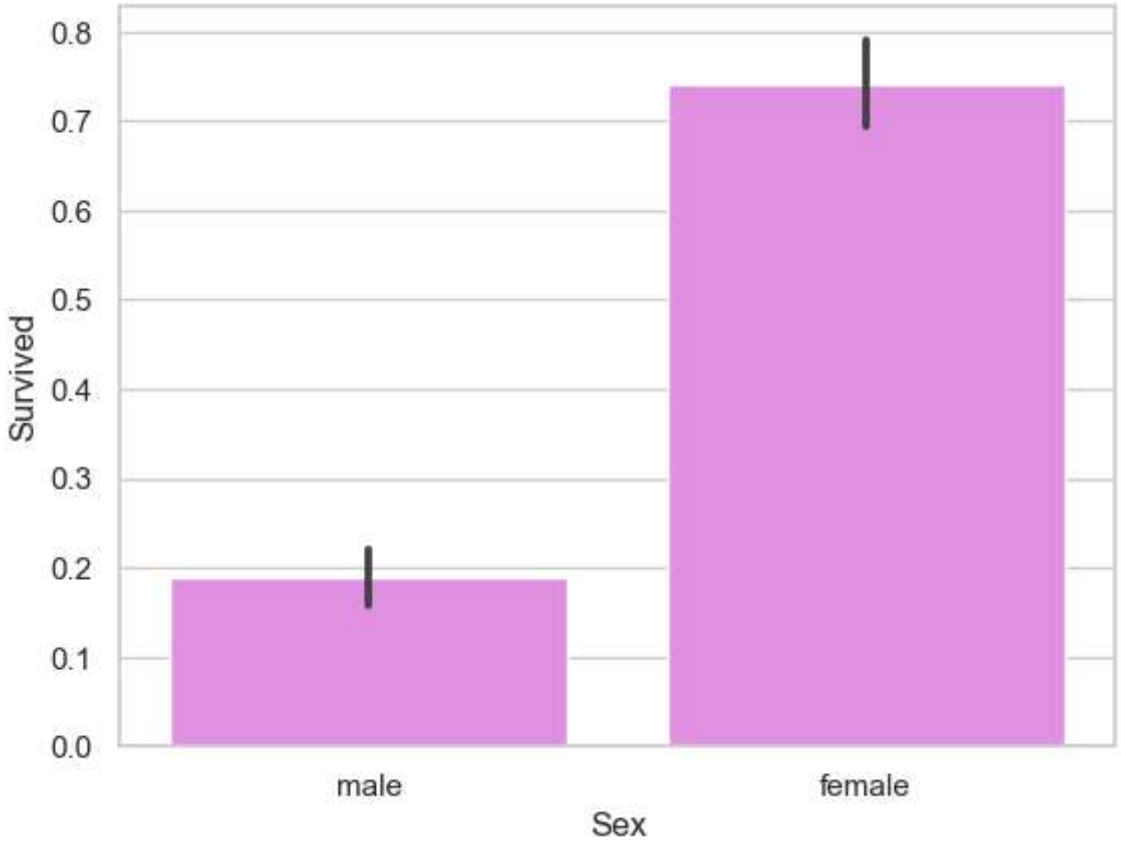
In [119]:
```
final_test['IsMinor']=np.where(final_test['Age']<=16, 1, 0)
print(final_test['IsMinor'])
```

```
0     0
1     0
2     0
3     0
4     0
5     1
6     0
7     0
8     0
9     0
10    0
11    0
12    0
13    0
14    0
15    0
16    0
17    0
18    0
```

In [120]:
```
sns.barplot(x='TravelAlone', y='Survived', data=final_train, color="Green")
plt.show()
```



In [121]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.barplot(x='Sex', y='Survived', data=train_df, color='violet')
plt.show()
```



Conclusion

Dataset we have taken is poor for linear model but with the smaller data works well with linear model

In [ ]: