problem statement

To find the best fit of the dataset

In [1]:
```
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
from sklearn.linear_model import Ridge,RidgeCV,Lasso
```

In [2]:
```
df=pd.read_csv(r"C:\Users\LENOVO\Downloads\fiat500_VehicleSelection_Dataset (2
df
```

Out[2]:

	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 [3]:
```
df=df[['age_in_days','km']]
df.columns=['Age','Km']
```

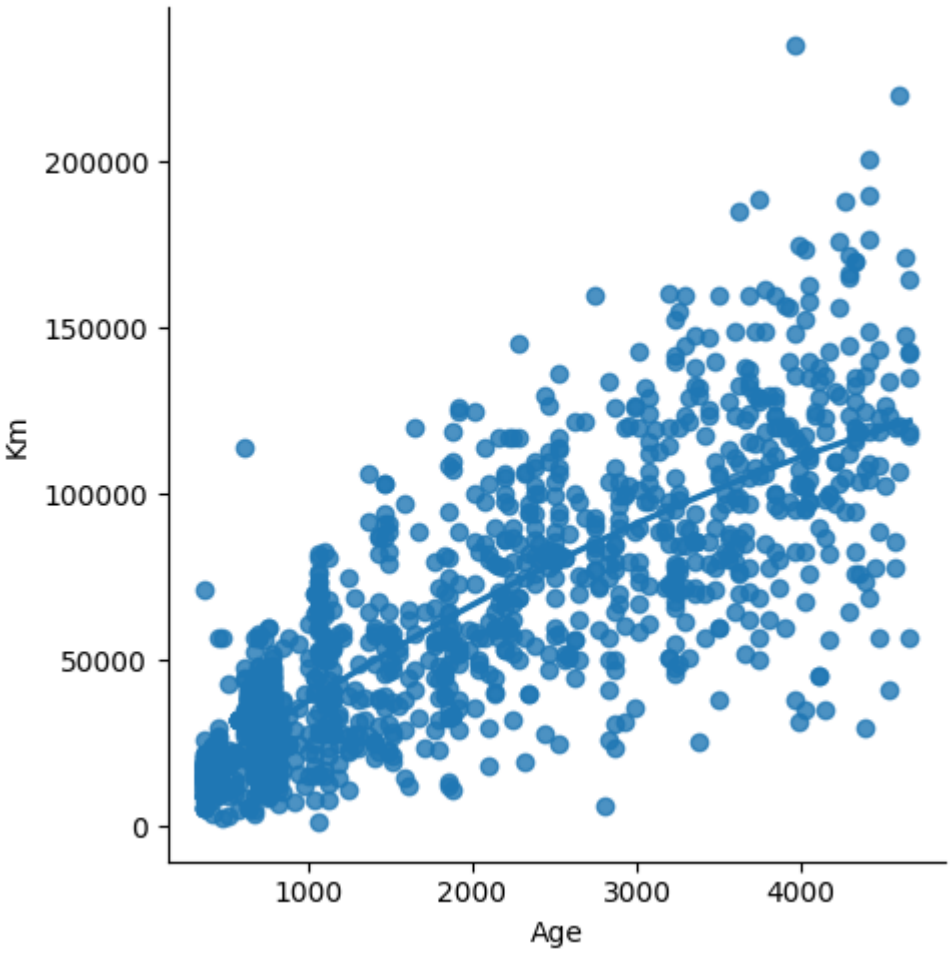
In [4]:
```
df.head(10)
```

Out[4]:

	Age	Km
0	882	25000
1	1186	32500
2	4658	142228
3	2739	160000
4	3074	106880
5	3623	70225
6	731	11600
7	1521	49076
8	4049	76000
9	3653	89000

In [5]:
```
sns.lmplot(x="Age",y="Km",data=df,order=2,ci=None)
```

Out[5]:
```
<seaborn.axisgrid.FacetGrid at 0x1f7def337d0>
```



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

Out[6]:

	Age	Km
count	1538.000000	1538.000000
mean	1650.980494	53396.011704
std	1289.522278	40046.830723
min	366.000000	1232.000000
25%	670.000000	20006.250000
50%	1035.000000	39031.000000
75%	2616.000000	79667.750000
max	4658.000000	235000.000000

In [7]:
```
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_7396\3337295870.py:1: SettingWit
hCopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [8]:
```
df.info()
```

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

In [9]:
```
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_7396\4116506308.py:1: SettingWit
hCopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [10]:
```
x=np.array(df['Age']).reshape(-1,1)
y=np.array(df['Km']).reshape(-1,1)
```

In [11]:
```
df.dropna(inplace=True)
```

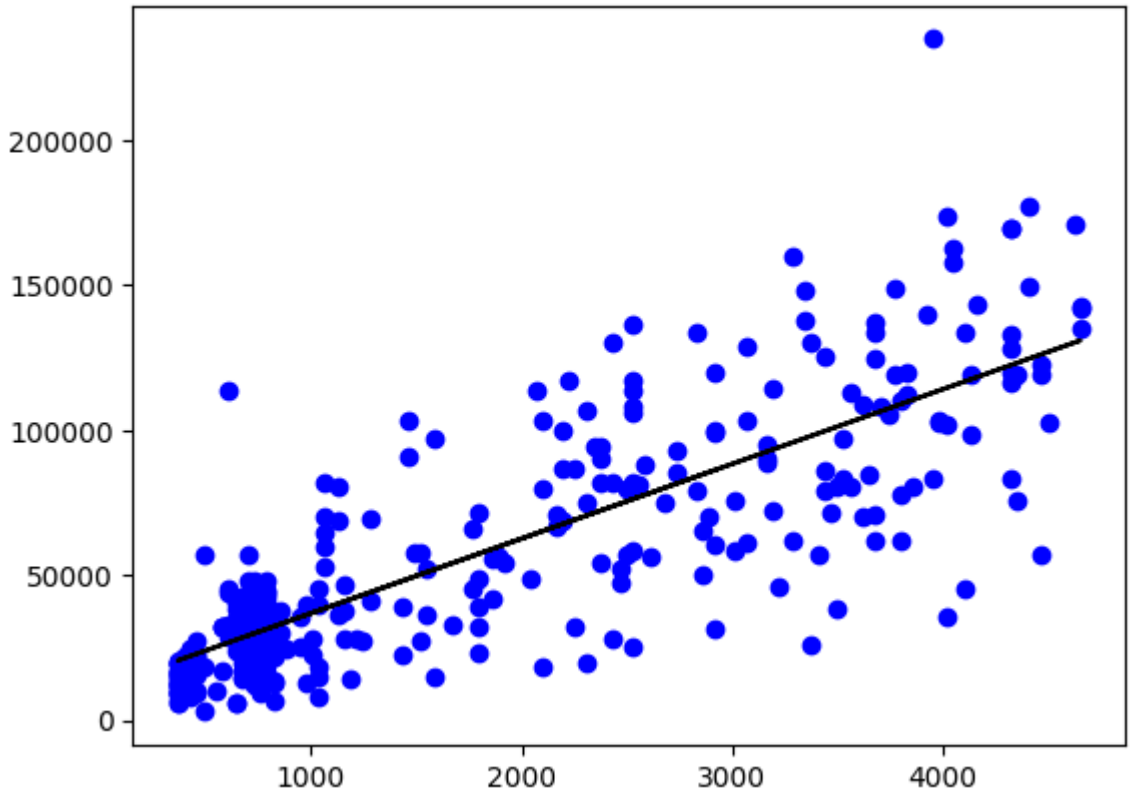
```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_7396\1379821321.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df.dropna(inplace=True)
```

In [12]:
```
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(regr.score(x_test,y_test))
```

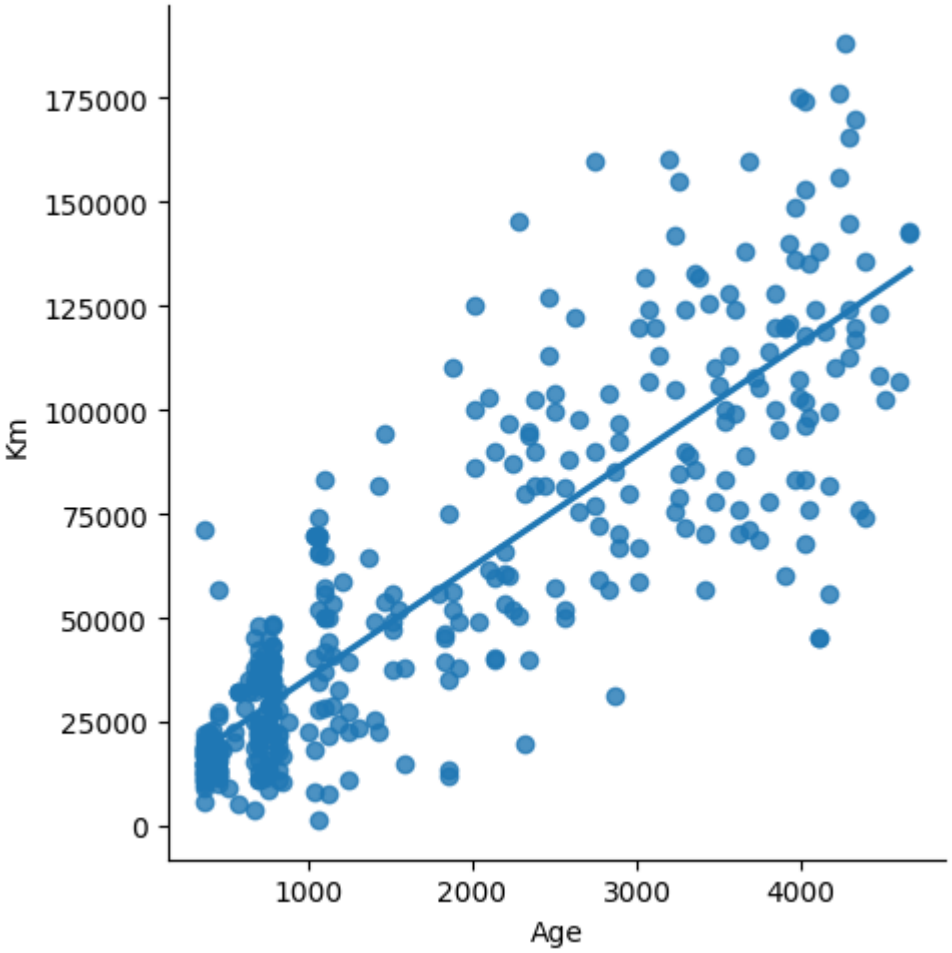
```
0.6989724893872842
```

In [13]:
```
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 [14]:
```
df400=df[:][:400]
sns.lmplot(x="Age",y="Km",data=df400,order=1,ci=None)
```

Out[14]: <seaborn.axisgrid.FacetGrid at 0x1f7e1a9e150>

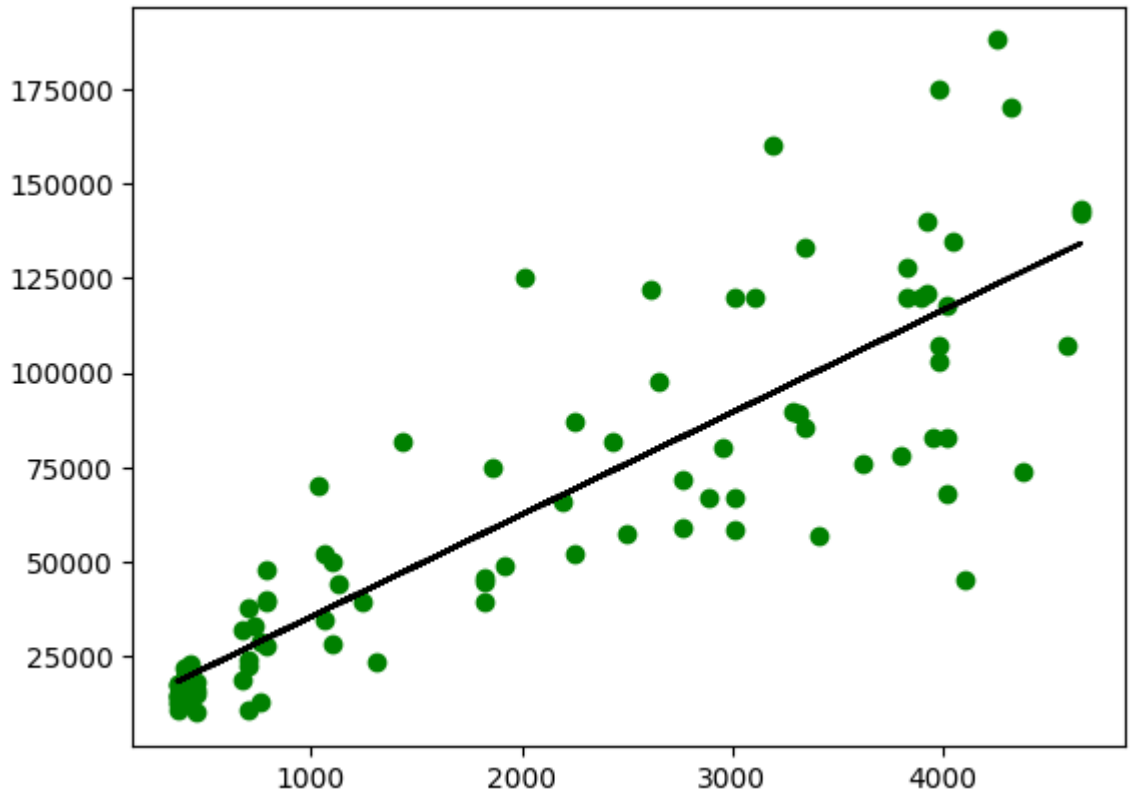


In [15]:
```
x=np.array(df400['Age']).reshape(-1,1)
y=np.array(df400['Km']).reshape(-1,1)
```

In [16]:
```
df400.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))
```

```
Regression: 0.748013678710749
```

In [17]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='g')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [18]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
```

Out[18]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [19]:
```
features=df.columns[0:8]
target=df.columns[-1]
```

In [20]:
```
x=df[features].values
y=df[target].values
```

In [21]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, rando
print(x_train.shape)
print(x_test.shape)
```

```
(1076, 2)
(462, 2)
```

In [22]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

In [24]:
```
regr=LinearRegression()
regr.fit(x_train,y_train)
train_score_a=regr.score(x_train, y_train)
test_score_a=regr.score(x_test,y_test)
print("\nLinear Regression Model:\n")
print(train_score_a)
print(test_score_a)
```

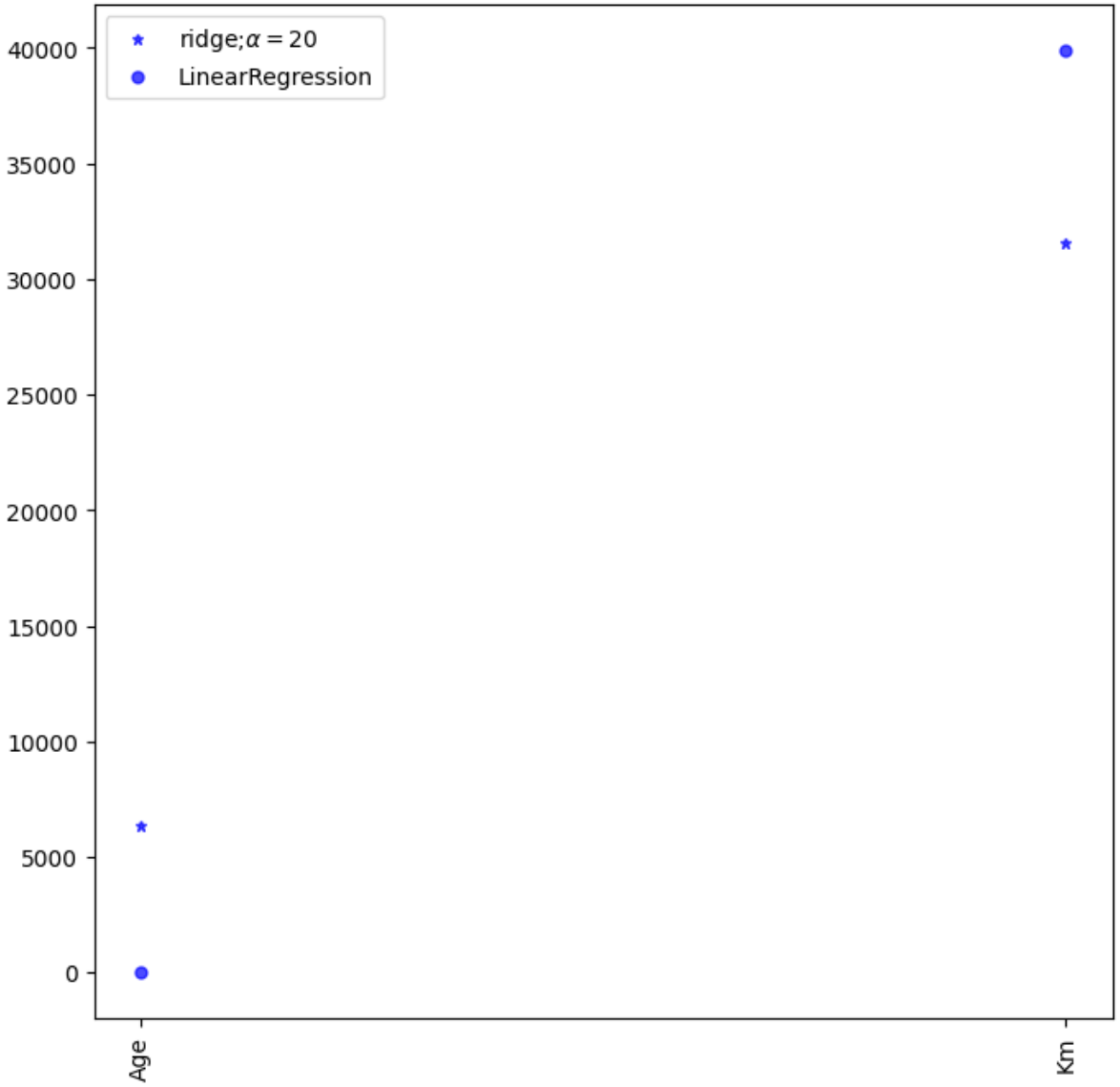
```
Linear Regression Model:

1.0
1.0
```

In [25]:
```
#ridge Regression
r=Ridge(alpha=100)
r.fit(x_train,y_train)
train_score_ridge=r.score(x_train,y_train)
test_score_ridge=r.score(x_test,y_test)
print(train_score_ridge)
print(test_score_ridge)
```

```
0.9870694292817765
0.9854347948093845
```

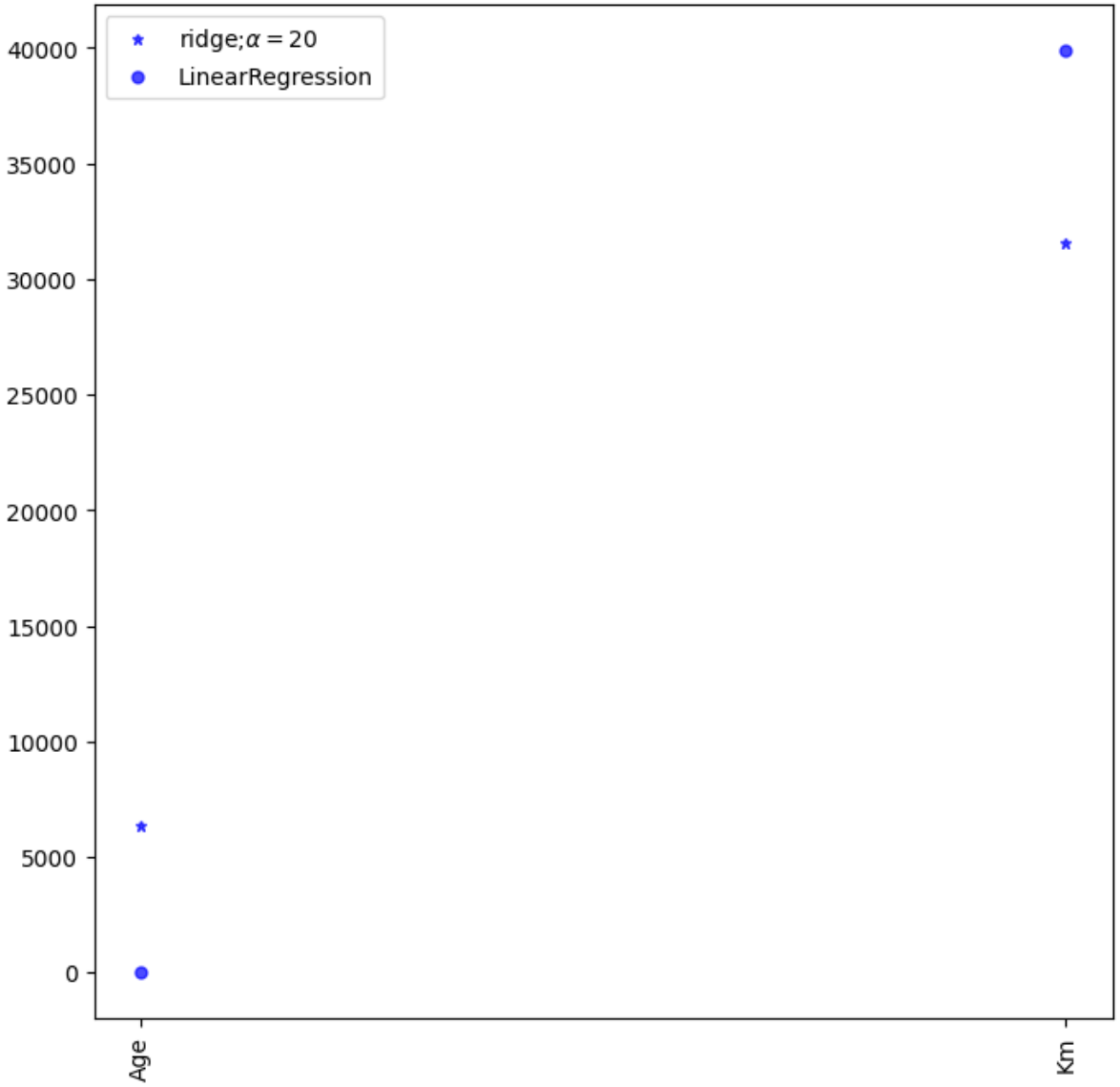
In [26]:
```
plt.figure(figsize=(8,8))
plt.plot(features,r.coef_,linestyle="None",alpha=0.7,marker='*',markersize=5,c
plt.plot(features,regr.coef_,alpha=0.7,linestyle="None",marker='o',markersize=
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [27]:
```
r=Ridge(alpha=100)
r.fit(x_train,y_train)
train_score_ridge=r.score(x_train,y_train)
test_score_ridge=r.score(x_test,y_test)
print(train_score_ridge)
print(test_score_ridge)
```

```
0.9870694292817765
0.9854347948093845
```

In [28]:
```
plt.figure(figsize=(8,8))
plt.plot(features,r.coef_,linestyle="None",alpha=0.7,markersize=5,color="blue"
plt.plot(features,regr.coef_,alpha=0.7,linestyle="None",markersize=5,color="bl
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

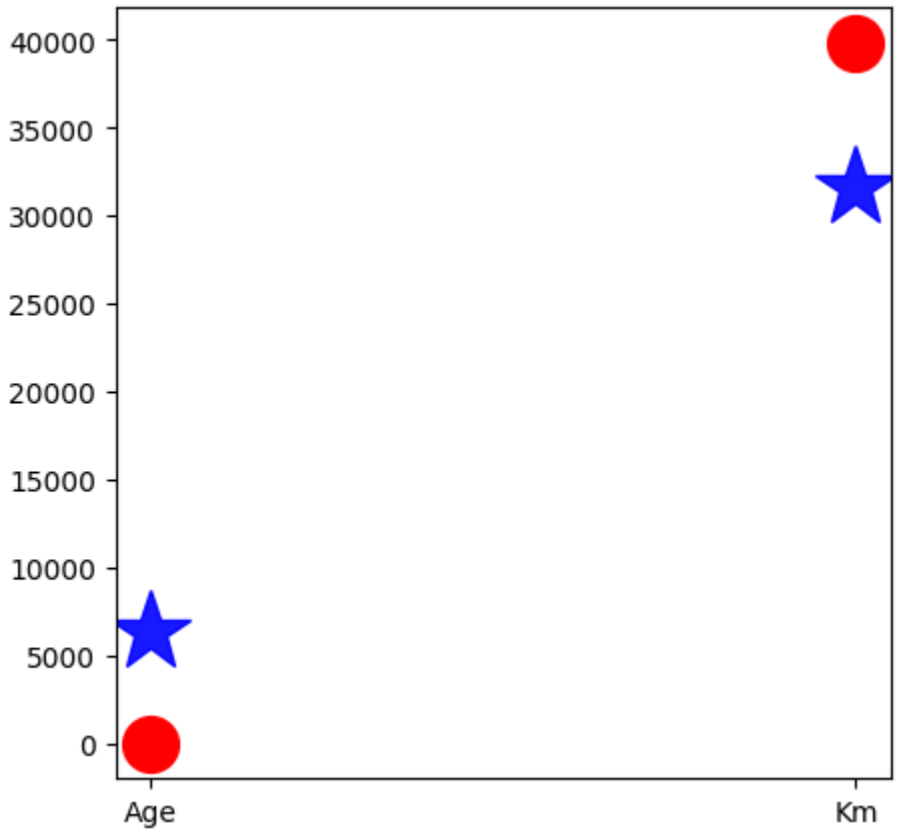


In [29]:
```
#Lasso Regression#
l=Lasso(alpha=10)
l.fit(x_train,y_train)
train_lasso=l.score(x_train,y_train)
test_lasso=l.score(x_test,y_test)
print(train_lasso)
print(test_lasso)
```

```
0.9999999359701485
0.9999999339169625
```

In [30]:
```
plt.figure(figsize=(5,5))
plt.plot(features,r.coef_,alpha=0.9,marker="*",markersize="30",label=r"ridge;$
plt.plot(features,l.coef_,alpha=1,marker="o",markersize=20,label="LinearRegres
```

Out[30]:
```
[<matplotlib.lines.Line2D at 0x1f7e1a65350>]
```



In [31]:
```
ridge_cv=RidgeCV(alphas=[0.0001,1.2,0.009,0.076]).fit(x_train,y_train)
```

In [32]:
```
print(ridge_cv.score(x_train,y_train))
print(ridge_cv.score(x_test,y_test))
```

```
0.9999999999999711
0.9999999999999679
```

In [33]:
```
from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(x,y)
print(regr.coef_)
print(regr.intercept_)
```

```
[1.59910693e-04 9.99995706e-01]
-0.03470170908985892
```

conclusion:

This is the best fit dataset since we got accuracy value is 98%.