Tesla Battery Survey

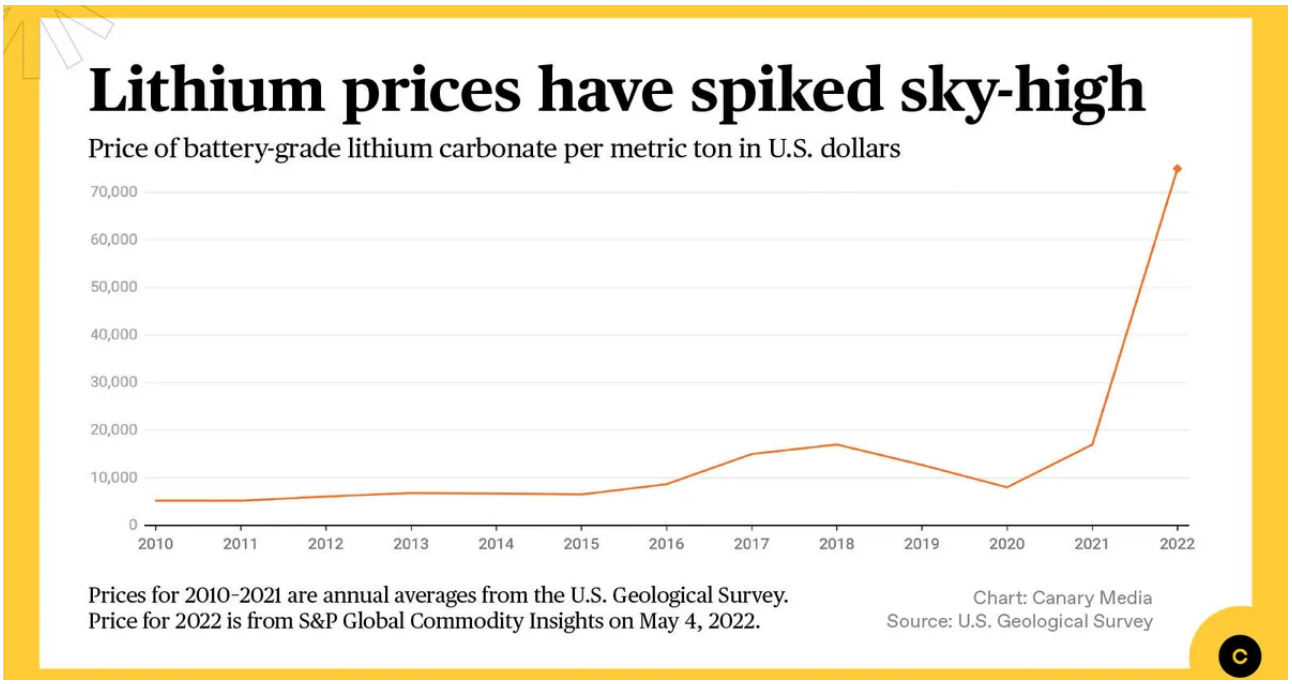
Understand range degradation of Tesla vehicles based on crowd-sourced statistics of owner-reported range estimates.

What is an electric vehicle battery?

An **electric vehicle battery** (**EVB**, also known as a **traction battery**) is a rechargeable battery used to power the electric motors of a battery electric vehicle (BEV) or hybrid electric vehicle (HEV). Typically lithium-ion batteries, they are specifically designed for high electric charge (or energy) capacity.

Most of today's all-electric vehicles and PHEVs use **lithium-ion batteries**, though the exact chemistry often varies from that of consumer electronics batteries. Research and development are ongoing to reduce their relatively high cost, extend their useful life, and address safety concerns in regard to overheating.

Currently 92% of production of lithium is coming from Australia(52.7%), Chile(21.5%), China(9.7%) and Argentina(8.3%).



Lithium prices were $8000 per metric ton till 2020 and it directly jumped to $20,000 per metric ton. It shows 400% of the raise. Cobalt and Nickel are also very costly.

The USA is a larger EV market and also an EV manufacturer followed by china.

Where traditional EV batteries have a liquidy, viscous lithium-based electrolyte whereas solid batteries used solid metal composition as their ion transport mechanism.
Solid state batteries are higher energy density.

Traditional lithium ion compositions used in EV battery pack stove about 114 Wh/lb or 250 Wh/kg Whereas one pound of solid state batteries could move tesla model 3 for 175-225 Wh/lb or 400-500 Wh/kg, the product is less costly for solid state battery.

What is Range?

Range is the distance your tesla can travel on a single charge. For battery-powered cars, we can also think of range as the amount of energy your battery has stored at a given point in time. Tesla drivers can choose to display range as either percentage remaining of an EPA(Environmental Protection Agency) rated mileage.

Displayed range in your tesla is adapted based on fixed EPA test data, not your personal driving patterns. It's natural for this to fluctuate due to the nature of battery technology and how the on board computers calculated range.

Diving Range

When the battery is fully charged, the range value represents the approximate number of miles that can be travelled in combined city and highway driving before the vehicle must be recharged.

Objective

Understand range degradation of Tesla vehicles based on crowd-sourced statistics of owner-reported range estimates.

In this report estimated range is reported by the user after full charge of the vehicle battery.

Data Preparation

1) Open the Excel spreadsheet included in the directory
2) Open a jupyter notebook and load sheets 'All entries' and 'Administrative'

   - Here I loaded both worksheet from excel file as I needed both worksheet during whole case study

```
In [3]:
# Imports (add)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

In [4]:
# Import data into Pandas DataFrame
filename = 'Tesla Battery Survey.xlsx'
# Load All entries worksheet
df = pd.read_excel(filename, sheet_name='All entries')
# Load Administrative worksheet
df_admin = pd.read_excel(filename, sheet_name='Administrative')
```

3) For each datapoint, compute *vehicle age* at point of measurement as difference between `Date you charged to 100%` and `Manufacture date`

1. What do you observe in this data? Any problems to address?

```
In [6]:
# compute *vehicle age* at point of measurement as difference
between `Date you charged to 100%` and `Manufacture date`
# using pandas' to_datetime function which converts each value in
the series into a 'datetime' object
df['Date you charged to 100%'] = pd.to_datetime(df['Date you
charged to 100%'])
df['Manufacture date'] = pd.to_datetime(df['Manufacture date'])
# Finding difference between two dates
df['vehicle_age'] = df['Date you charged to 100%'] -
df['Manufacture date']
# Automatically output will be in days but I converted into years
# dt.total_seconds is a property of timedelta object and it will
return total number in seconds
df['vehicle_age'] = df['vehicle_age'].dt.total_seconds() / (365 *
24 * 3600)
df['vehicle_age'] = np.round(df['vehicle_age'], 2)
# Printing three columns from dataframe
print(df[['Date you charged to 100%', 'Manufacture date',
'vehicle_age']])
```

```
    Date you charged to 100% Manufacture date  vehicle_age
0                 2015-06-10       2015-04-14         0.16
1                 2016-12-24       2016-06-20         0.51
2                 2018-01-18       2016-08-22         1.41
3                 2020-01-11       2016-05-04         3.69
4                 2017-08-23       2013-07-20         4.10
...                      ...              ...          ...
1530              2015-08-11       2015-07-07         0.10
1531              2015-07-29       2015-03-05         0.40
1532              2014-10-04       2013-12-10         0.82
1533              2015-01-21       2013-12-10         1.12
1534              2015-05-05       2013-12-10         1.40

[1535 rows x 3 columns]
```

```
In [7]:
# Let's observe data
# Average value of vehicle age
mean = np.mean(df['vehicle_age'])
print(mean)
median = np.median(df['vehicle_age'])
print(median)
1.651299804049638
nan
```

As here we can see we got a median as nan because some of the users did not enter the manufacture date. Let’s fill out those values with mean.

[Here I am using the Imputation method in which: Missing data can be filled in with statistical methods such as mean, median, or mode imputation. This method can be used if the missing data is missing at random and does not have a significant impact on the overall analysis.]

```
In [8]:
# As here we can see we got median as nan because some of the
vehicle age is nan
# because Customer did not fill up manufacture date so added mean
instead of those NAN values
df['vehicle_age'].fillna(mean, inplace=True)
# again find median
median = np.median(df['vehicle_age'])
print(median)
1.37
```

4) For each datapoint, use K column in sheet `Administrative` to assign *usable Wh capacity* for each model/configuration and enhance the dataset (create new column) with this parameter

```
In [9]:
# above we already load administrative file as df_admin
# I create dictionary for model and Wh because Wh capacity is
starting from K4
usable_Wh_capacity_dict = dict(zip(df_admin.iloc[2:, 5],
df_admin.iloc[2:, 10]))
print(usable_Wh_capacity_dict)
{'Unspecified 85 kWh': 77500, 'Model S 60': 58500, 'Model S 60D':
58500, 'Model S 70': 68800, 'Model S 70D': 68800, 'Model S 75':
73200, 'Model S 75D': 73200, 'Model S 85': 77500, 'Model S P85':
77500, 'Model S P85+': 77500, 'Model S 85D': 77500, 'Model S
P85D': 77500, 'Model S 90': 81800, 'Model S 90D 2015': 81800,
'Model S 90D': 81800, 'Model S P90D': 81800, 'Model S 100D':
98400, 'Model S P100D': 98400, 'Model X 60D': 62400, 'Model X
70D': 68800, 'Model X 75D': 73200, 'Model X 90D': 81800, 'Model X
P90D': 81800, 'Model X 100D': 98400, 'Model X P100D': 98400,
'Model 3 LR': 78270, 'Model 3 LR AWD': 78270, 'Model 3 P': 78270,
'Model 3 MR': 63810, 'Model 3 SR': 47475, 'Model 3 SR+': 52750,
nan: nan}

In [10]:
# here df is an all entries dataframe
# Define a function to retrieve the usable Wh capacity based on
the Vehicle model
def get_usable_Wh_capacity(model_configuration):
    return usable_Wh_capacity_dict.get(model_configuration, 0)
```

```
# Apply above function to the "Vehicle Model" column in the All
entries dataframe to create a new column
# Vehicle model is a column in df which is define user is using
which model
df['Usable Wh Capacity'] = df['Vehicle
model'].apply(get_usable_Wh_capacity)
In [13]:
# Checking info for df so I can see the columns are added or not
df.info()
```

What is cumulative energy and how can we measure it.

X-Axis = miles
Y-Axis = Wh/mi showing energy (In some regions it shows Wh/km)
Multimodal distribution
Charts show what amount of energy we are using during our journey.
Projected range will show the number of miles you have left in a battery
Wh/mi means tesla using that amount of particular energy in 1 miles or 1 kilometres, also call it lifetime_avg which is given in our data

Lower the value of Wh/mi which means we are using less energy or if it is <0 which means we are putting energy back to battery via recharging battery or going down through hills. If the value of Wh/mi is more than 300 then we are using more energy.

If the graph shows instant range or less miles like above in the chart we will see the projected range up and down very rapidly. For accurate measure it is better to see the average range.

In this graph we can see the average range for 30 miles. Also we can notice two horizontal lines. One is a solid line which represents the less EPR miles. And the dash line represents a more reliable and accurate projected range in miles. Dash line move up and move down based on how much energy we are using. If it is above solid line we have less miles then EPR (Estimated projected range) and if it is below the solid line which means more miles then EPR or the same as it's showing.

In [14]:

```
# To count cumulated energy we need to multiply lifetime avg to current
mileage of customer
```

```
# as we see above Lifetime average [Wh/mi or Wh/km]? is available in df
but some user put it in miles and some of them put it in km
# So we will convert all Lifetime average [Wh/mi or Wh/km]? into
lifetime_avg_km in new column
# Create a new column "lifetime_avg_km"
df["lifetime_avg_km"] = 0
# Loop through the dataframe rows
for index, row in df.iterrows():
    # Check if the "range unit" column is "mi"
    if row["Range unit"] == "mi. My car displays range in miles and I
will enter all range numbers in miles.":
        # Convert the "lifetime avg" column from miles to kilometres
        df.at[index, "lifetime_avg_km"] = row["Lifetime average [Wh/mi
or Wh/km]?"] * 1.60934
    # Check if the "range unit" column is "km"
    elif row["Range unit"] == "km. My car displays range in kilometres
and I will enter all range numbers in kilometres.":
        # Keep the "lifetime avg" column in kilometres
        df.at[index, "lifetime_avg_km"] = row["Lifetime average [Wh/mi
or Wh/km]?"]
df['lifetime_avg_km'] = np.round(df['lifetime_avg_km'], 2)

In [16]:
# As we can see here there are some null values in lifetime_avg_km
# So let's count mean and median and let's see what we can fill into
the null values so here I used method Imputation
# Average value of lifetime_avg_km
mean = np.mean(df['lifetime_avg_km'])
print(mean)
median = np.median(df['lifetime_avg_km'])
print(median)
# Here I did not count and use mode because lifetime_avg is depend on
person to person
#may be some of users are driving with very high energy so it is better
to use mean
283.26822161422706
nan

In [17]:
# as we can see got median as nan so let's fill null values with mean
df['lifetime_avg_km'].fillna(mean, inplace=True)
median = np.median(df['lifetime_avg_km'])
print(median)
223.0

In [20]:
# Now let's count cumulated energy
df["cumulated_energy"] = df["lifetime_avg_km"] * df["Mileage [km]"]
In [21]:
#Now let's count equivalent full cycles
```

```
df['equivalent_full_cycles'] = df['cumulated_energy'] / df['Usable Wh Capacity']
```

Analysis

1) How many unique vehicles/users are represented in the dataset?

```
In [28]:
# Task 1 How many unique vehicles/users are represented in the dataset?
# count total number of unique vehicle model
unique_vehicles = df['Vehicle model'].unique()
print("Number of unique vehicles:", len(unique_vehicles))
#Count all unique user
unique_usernames = df['Username'].unique()
print("Number of unique usernames:", len(unique_usernames))
# print those all unique vehicle
print("Unique vehicle models:", unique_vehicles)
Number of unique vehicles: 28
Number of unique usernames: 610
Unique vehicle models: ['Model S 85' 'Model S 90D' 'Model S P85'
'Model S 75' 'Model S P85D' 'Unspecified 85 kWh' 'Model S 85D'
'Model S 70D' 'Model S 70' 'Model S 75D' 'Model X 100D' 'Model 3
LR' 'Model X 75D' 'Model S P90D' 'Model S P85+' 'Model 3 P'
'Model S 60' 'Model 3 SR+' 'Model X 60D' 'Model S 60D' 'Model X
90D' 'Model S 90' 'Model S P100D' 'Model S 100D' 'Model 3 LR AWD'
'Model S 90D 2015' 'Model 3 SR' 'Model X P90D']
```

2) What is the count of different reported models (group by Model S/X/3/Other - not detailed configuration)? Example: Model S: 1000, Model X: 1000, Model 3: 50, Other: 30.

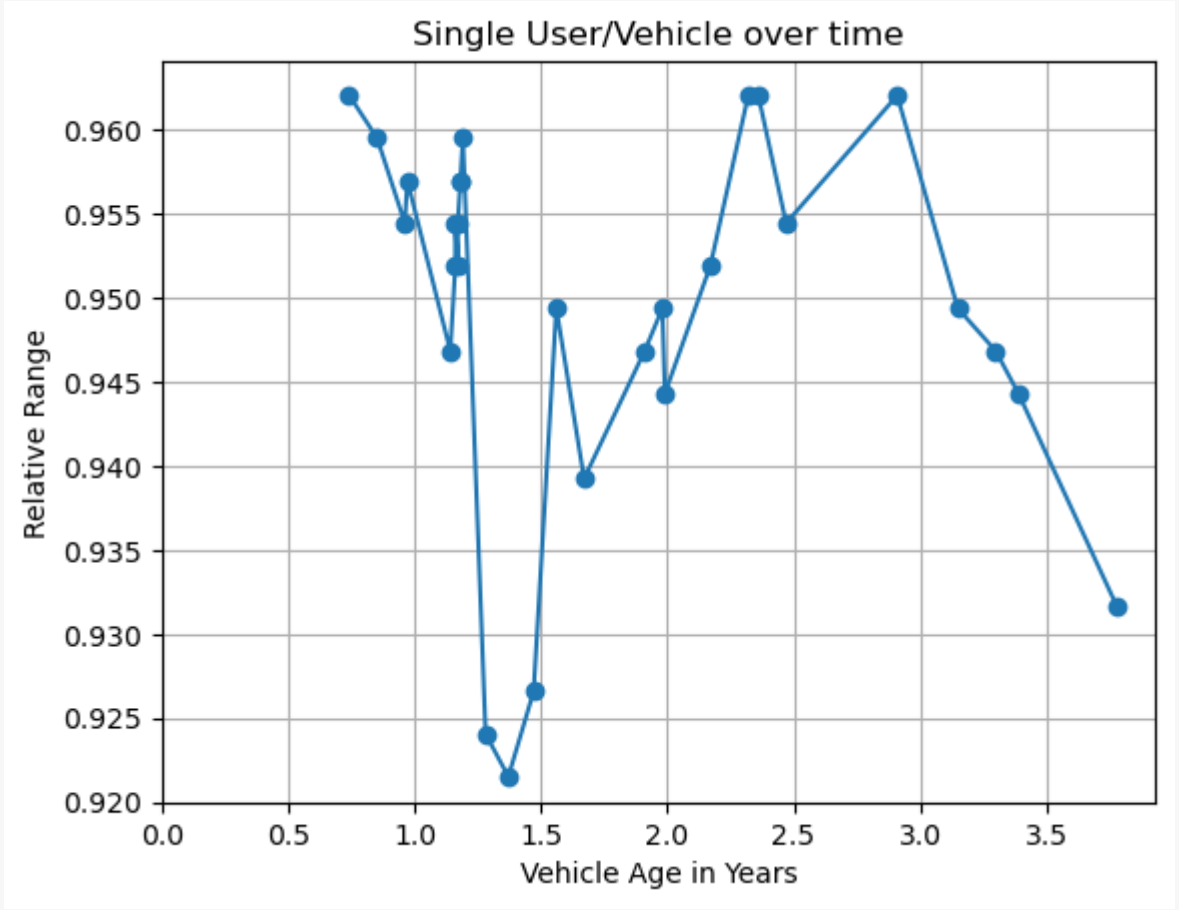
```
In [29]:
#2.  What is the count of different reported models (group by Model S/X/3/Other - not detailed configuration)?
user_counts = df.groupby('Vehicle model')['Username'].nunique()
print(user_counts)
Vehicle model
Model 3 LR              11
Model 3 LR AWD           8
Model 3 P               11
Model 3 SR               2
Model 3 SR+              7
Model S 100D             6
Model S 60              34
Model S 60D              6
```

```
Model S 70              8
Model S 70D            29
Model S 75             18
Model S 75D            27
Model S 85            163
Model S 85D            54
Model S 90              3
Model S 90D            53
Model S 90D 2015        7
Model S P100D           6
Model S P85            77
Model S P85+           16
Model S P85D           26
Model S P90D            6
Model X 100D           10
Model X 60D             1
Model X 75D             6
Model X 90D            10
Model X P90D            1
Unspecified 85 kWh     20
Name: Username, dtype: int64
```

3) Create a line plot with Relative Range against vehicle age in years (`z_actual` vs `x`) for the vehicle with the largest number of datapoints recorded.

```
In [31]:
# Create line plot with Relative Range against vehicle age in
years (`z_actual` vs `x`)
# for the vehicle with largest number of datapoints recorded.
most_common_vehicle = df['Vehicle model'].value_counts().index[0]
print(most_common_vehicle)
# Filter the dataframe to get only the rows where vehicle model
is the most common vehicle
df_filtered = df[df['Vehicle model'] == most_common_vehicle]
# Get the username column for the filtered dataframe
username_counts = df_filtered['Username'].value_counts()
# Get the most common username for the filtered dataframe
most_used_username = username_counts.index[0]
print(most_used_username)
# Filter the dataframe again to get only the rows where username
is the most common username
df_filtered = df_filtered[df_filtered['Username'] ==
most_used_username]
z_actual = df_filtered['z_actual']
vehicle_age = df_filtered['vehicle_age']
Model S 85
SSc3010
```

```
In [32]:
# Plot the relative range against vehicle age in years
plt.plot(vehicle_age, z_actual, '-o')
plt.xlabel('Vehicle Age in Years')
plt.ylabel('Relative Range')
plt.title('Single User/Vehicle over time')
plt.ylim(0.920, None) # Set the y-axis lower limit
plt.xlim(0, None)
plt.grid() # Add a grid to the plot
plt.show()
```

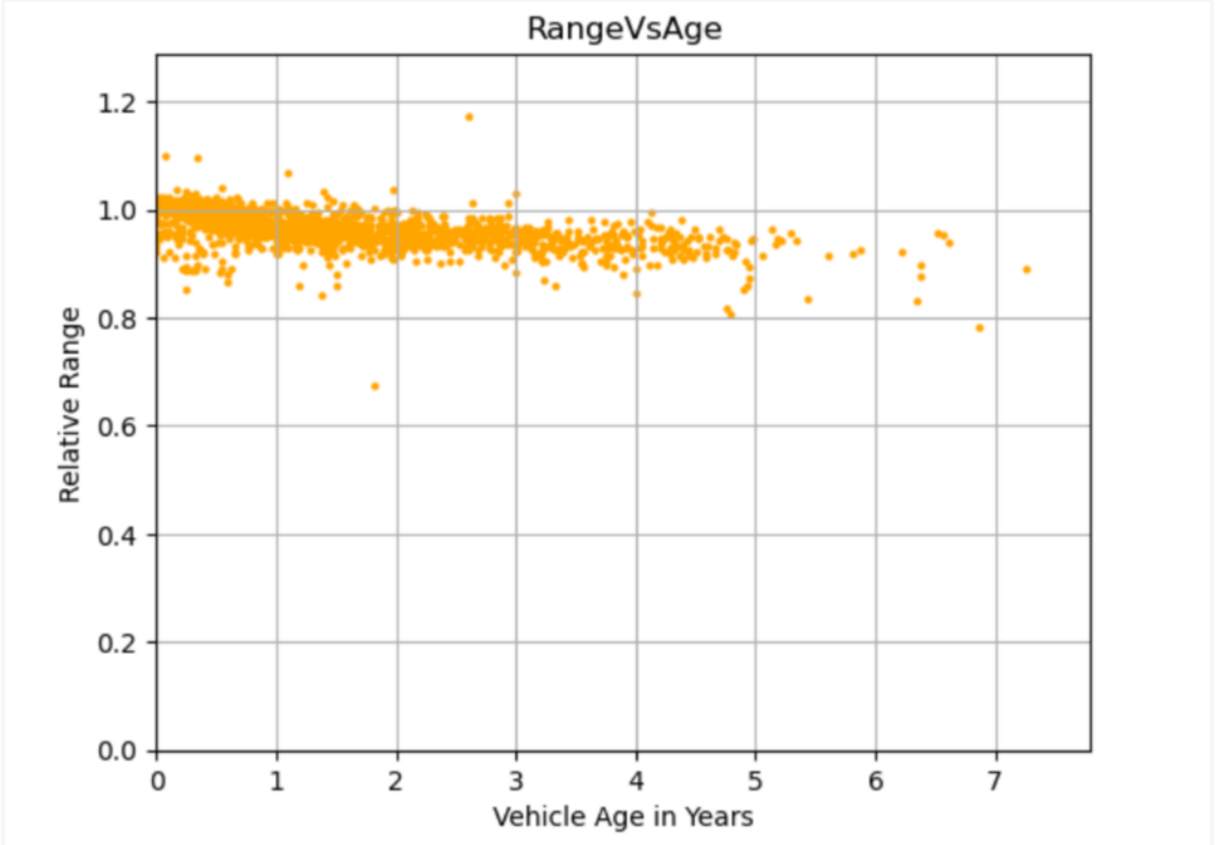


4) Create a scatter plot with Relative Range against vehicle age in years (`z_actual` vs `x`) for all data points (all vehicles).

 # As we all remember we find vehicle age. We also got some ages in minus because customers provide wrong details. So here I exclude all those negative details. Basicallyl use "exclusions" method refers to the process of removing certain data points or entries from the dataset. This is done when the data points in question do not meet certain criteria, or when they contain errors or missing values. The process of excluding data points is typically used to clean up the dataset and improve the accuracy of the analysis.

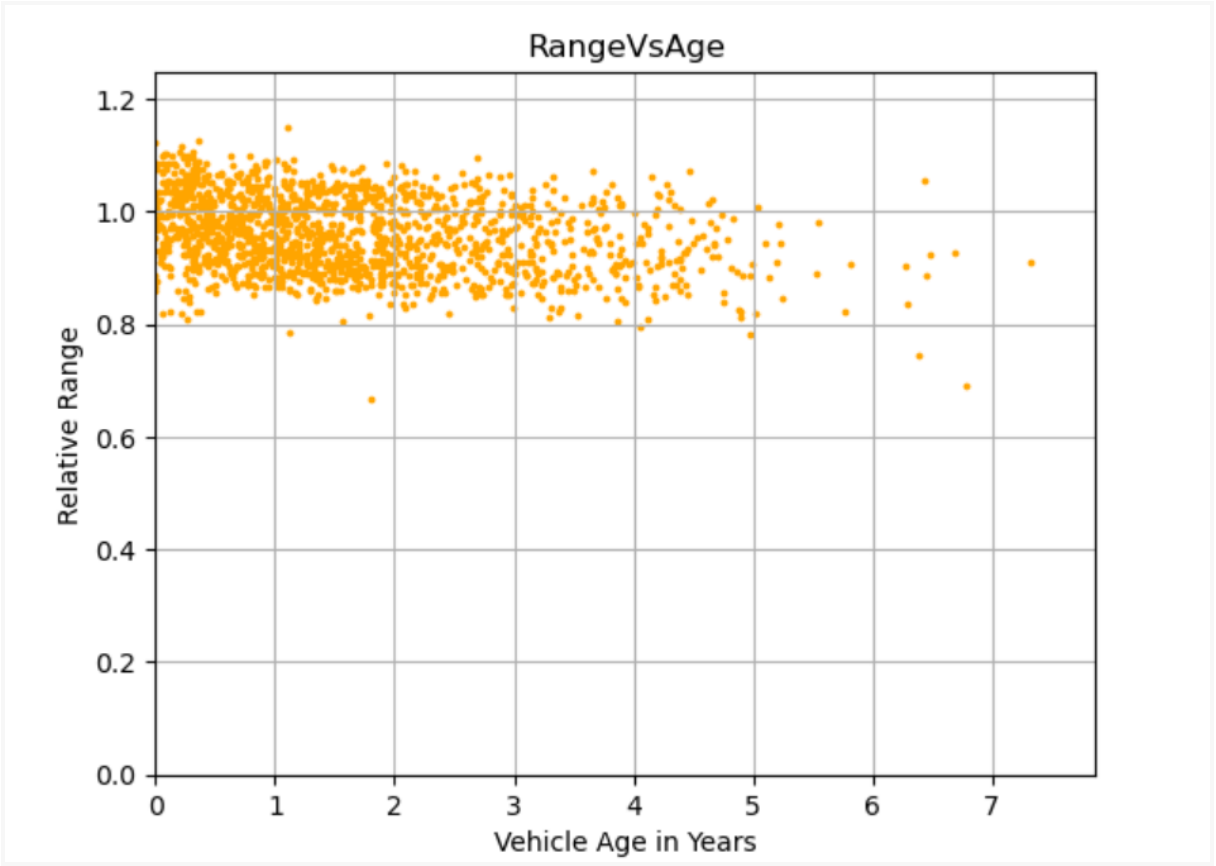
Here we find two continuous variable vehicle age and relative range so we used a scatter plot to see the relation between these two variables. Below scatter plot shows slightly weak negative scatter plot.

In [43]:
```
# Create a scatter plot with Relative Range against vehicle age
in years (`z_actual` vs `x`) for all data points (all vehicles).
# Plot the relative range against vehicle age in years
plt.scatter(df['vehicle_age'], df['z_actual'],  color='orange',
s=4)
plt.xlabel('Vehicle Age in Years')
plt.ylabel('Relative Range')
plt.title('RangeVsAge')
plt.ylim(0, None) # Set the y-axis lower limit
plt.xlim(0, None)
plt.grid() # Add a grid to the plot
plt.show()
```



In [45]:
```
df['vehicle_age'] += np.random.uniform(low=-0.1, high=0.1,
size=df.shape[0])
df['z_actual'] += np.random.uniform(low=-0.1, high=0.1,
size=df.shape[0])
```

```
plt.scatter(df['vehicle_age'], df['z_actual'], color='orange',
s=3)
plt.xlabel('Vehicle Age in Years')
plt.ylabel('Relative Range')
plt.title('RangeVsAge')
plt.ylim(0, None) # Set the y-axis lower limit to 0
plt.xlim(0, None)
plt.grid() # Add a grid to the plot
plt.show()
```

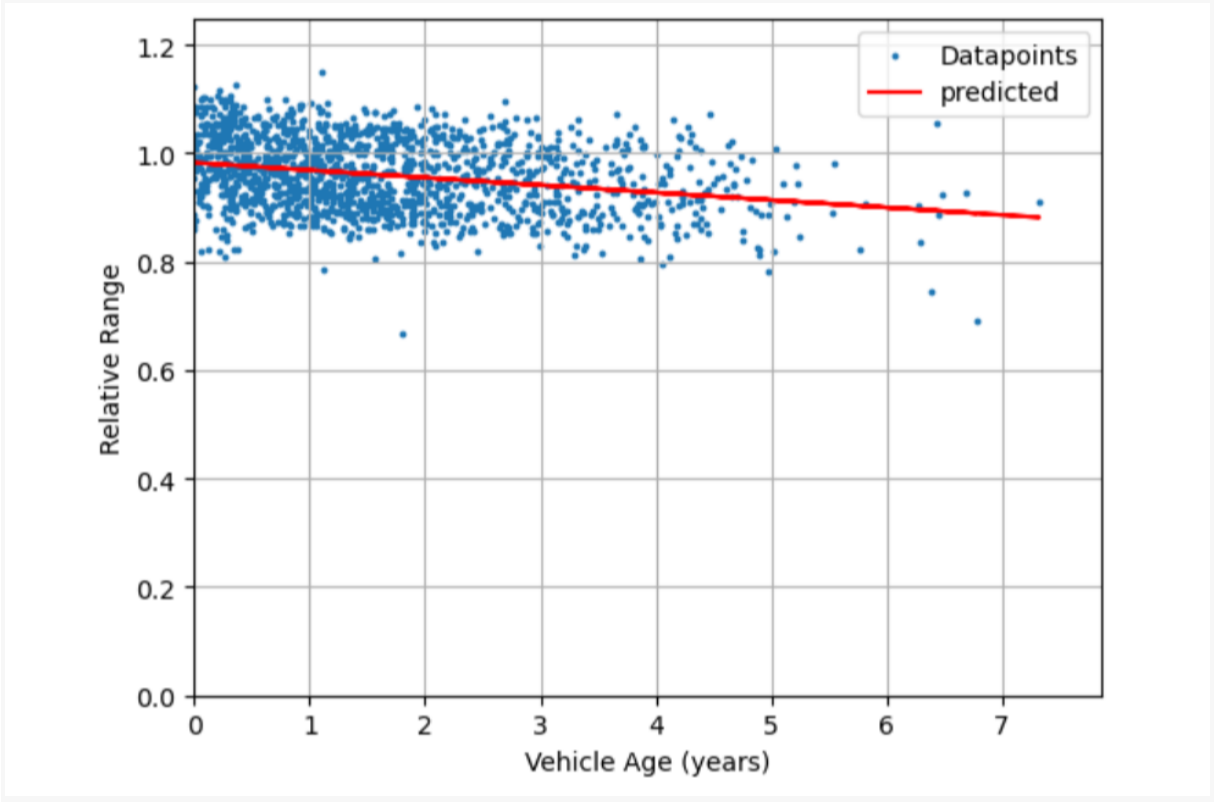


5) Create a linear model (fit a line) `z = f(x) = A + B*x` (optimize by least square error) to estimate Relative Range from vehicle age. You are encouraged to make use of existing open source code to help your optimization.

   1. Plot Relative Range model evaluation (`z_pred` vs `x`) in the same figure as corresponding datapoints scatter plot (`z_actual` vs `x`) so that the line goes through the datapoints
   2. Plot distribution (histogram plot) of model estimation errors (`z_actual` - `z_pred`)
   3. Calculate standard error (standard deviation) of the Relative Range estimates

In [48]:

```
from sklearn import datasets, linear_model
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from scipy import stats
X = sm.add_constant(df['vehicle_age'])
#Ordinary least square method
est = sm.OLS(df['z_actual'], X).fit() # this is a OLS object
z_pred = est.predict(X) # use the predict method of the object
df['z_pred'] = z_pred
# Plot the model evaluation and scatter plot of datapoints
plt.scatter(df['vehicle_age'], df['z_actual'],
label='Datapoints', s=3)
plt.plot(X, df['z_pred'], '-r', label='predicted')
plt.xlabel('Vehicle Age (years)')
plt.ylabel('Relative Range')
plt.ylim(0, None)
plt.xlim(0, None)
plt.legend()
plt.grid()
plt.legend()
plt.show()
```

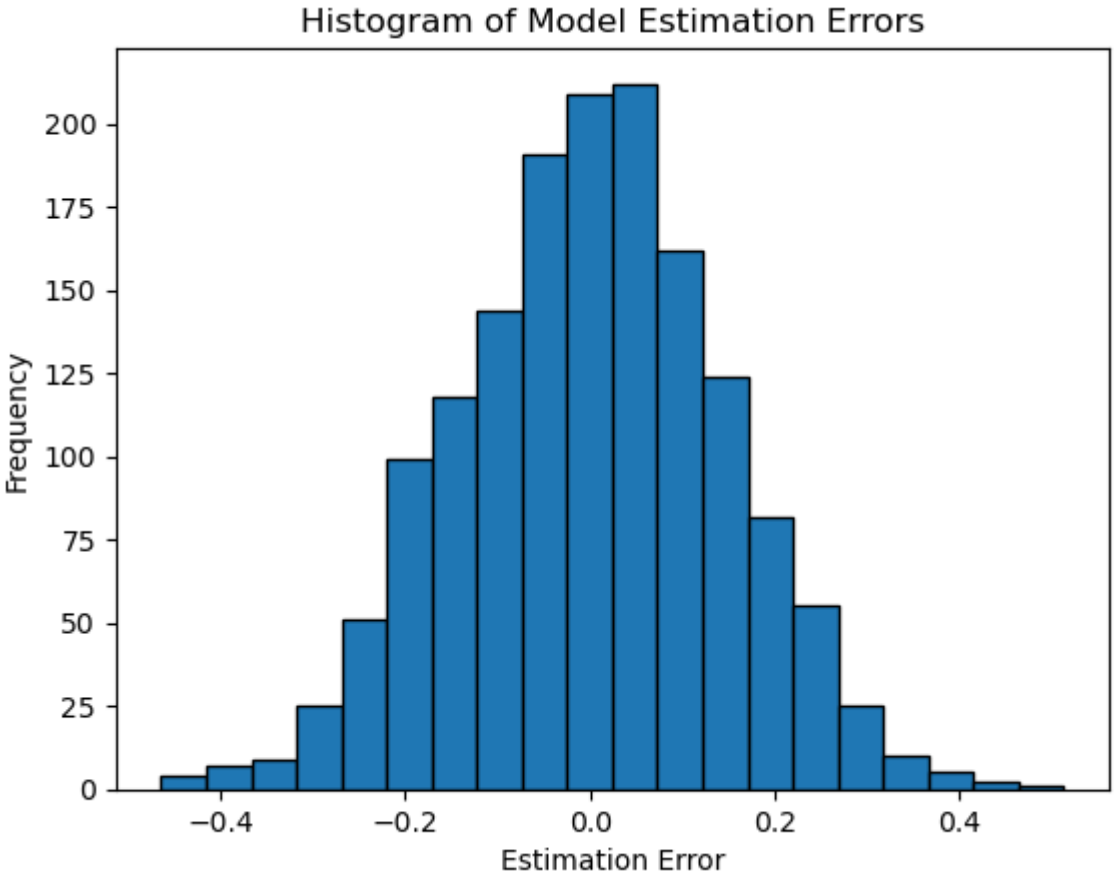


In [49]:

```
import matplotlib.pyplot as plt
# Calculate the estimation errors
```

```
error = df['z_actual'] - df['z_pred']
# Plot the histogram of the errors
plt.hist(error, bins=20, edgecolor='black')
plt.xlabel('Estimation Error')
plt.ylabel('Frequency')
plt.title('Histogram of Model Estimation Errors')
#plt.ylim(0, None)
#plt.xlim(0, None)
plt.show()
```

Based on below histogram we can say it is symmetric histogram because of normal distribution. It is based on bell curve and also called mesokurtic means many extreme values.



In [52]:

```
import numpy as np
# Calculate the estimation errors
error = df['z_actual'] - df['z_pred']
# Calculate the standard deviation of the errors
std_error = np.std(error)
print('Standard deviation of the estimation errors:', std_error)
```

```
Standard deviation of the estimation errors: 0.14389333065227883
```

Conclude:

We can conclude that there is a relationship between the age of Tesla vehicles and their relative range degradation.
The linear model (OLS) we fit shows that as the vehicle age increases, the relative range decreases.
The scatter plot of the data points and the model evaluation supports this conclusion, showing that the line fits through the data points, indicating that there is a relationship between the two variables.
The histogram of the model estimation errors shows a normal distribution, with a standard deviation of 0.14. This means that the model's predictions have a deviation of around 0.14 units from the actual values.
In conclusion, we can say that the relative range degradation of Tesla vehicles is predictable based on their age, and the linear model we fit provides a good estimate of the degradation based on crowd-sourced statistics of owner-reported range estimates.

Based on the conclusion we can advise following points:

1) **Regular maintenance**: Regular maintenance of the Tesla vehicles can help to reduce the range degradation and improve the overall performance of the vehicle.
2) **Age-based replacement:** Based on the vehicle age, we can advise the owners to replace their vehicles with a new one to ensure better performance and range.
3) **Monitor the range:** we can advise the owners to regularly monitor their vehicle's range and report any noticeable decrease to the service centre.
4) **Use data-driven decisions**: we can make data-driven decisions by utilizing the model developed in this analysis and predict the range degradation for a particular vehicle based on its age. This can help to make proactive decisions and plan maintenance and replacement schedules.