databricks Powering the Road_ A comprehensive Analysis on Electric Vehicles (1)

(https://databricks.com)

```
#read dataset
df1 = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/mgrover3@gmu.edu/Electric_Ve
```

```
#display the dataframe
display(df1)
```

Table

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make
1	5UXKT0C59G	Yakima	Zillah	WA	98953	2016	BMW
2	5YJ3E1EA2J	Snohomish	Edmonds	WA	98020	2018	TESLA
3	1G1RE6E4XE	Kitsap	Port Orchard	WA	98367	2014	CHEVROLET
4	2C4RC1L76M	Skagit	Bow	WA	98232	2021	CHRYSLER
5	5YJ3E1EA2J	Thurston	Olympia	WA	98513	2018	TESLA
6	WA1E2BFY8N	Snohomish	Snohomish	WA	98296	2022	AUDI

7,404 rows | Truncated data

```
#display summary for the dataframe.
display(df1.summary())
```

Table

	summary	VIN (1-10)	County	City	State	Postal Code	Model Year	M
1	count	173533	173528	173528	173533	173528	173533	17
2	mean	null	null	null	null	98174.74609861233	2020.4353523537309	nu
3	stddev	null	null	null	null	2411.1096851358934	2.99444160055937	nu
4	min	1C4JJXN60P	Ada	Aberdeen	AE	01545	1997	Al
5	25%	null	null	null	null	98052.0	2018.0	nu
6	50%	null	null	null	null	98122.0	2022.0	nu

8 rows

```
from pyspark.sql.functions import when, count, col,mean

#count number of null values in each column of DataFrame
df1.select([count(when(col(c).isNull(), c)).alias(c) for c in df1.columns]).show()
```

```
+----------+------+----+-----+-----------+----------+----+-----+---------------------+-----------------------------------
--------------+--------------+---------+--------------------+--------------+-----------------+----------------+-----------
------+
|VIN (1-10)|County|City|State|Postal Code|Model Year|Make|Model|Electric Vehicle Type|Clean Alternative Fuel Vehicle (CAF
V) Eligibility|Electric Range|Base MSRP|Legislative District|DOL Vehicle ID|Vehicle Location|Electric Utility|2020 Census
Tract|
+----------+------+----+-----+-----------+----------+----+-----+---------------------+-----------------------------------
--------------+--------------+---------+--------------------+--------------+-----------------+----------------+-----------
------+
|         0|     5|   5|    0|          5|         0|   0|    0|                    0|
0|             1|        1|                 376|             0|               10|               5|                5|
+----------+------+----+-----+-----------+----------+----+-----+---------------------+-----------------------------------
--------------+--------------+---------+--------------------+--------------+-----------------+----------------+-----------
------+
```

```
# Calculate mode of the categorical column
mode_value = df1.groupBy("County").count().orderBy(col("count").desc()).select("County").limit(1).collect()[0][0]

# Replace null values with mode
imputed_df = df1.withColumn("County", when(col("County").isNull(), mode_value).otherwise(col("County")))
mode_value = df1.groupBy("City").count().orderBy(col("count").desc()).select("City").limit(1).collect()[0][0]
imputed_df = imputed_df.withColumn("City", when(col("City").isNull(), mode_value).otherwise(col("City")))
mode_value = df1.groupBy("Postal Code").count().orderBy(col("count").desc()).select("Postal Code").limit(1).collect()[0][
imputed_df = imputed_df.withColumn("Postal Code", when(col("Postal Code").isNull(), mode_value).otherwise(col("Postal Cod
mode_value = df1.groupBy("Legislative District").count().orderBy(col("count").desc()).select("Legislative District").limi
imputed_df = imputed_df.withColumn("Legislative District", when(col("Legislative District").isNull(), mode_value).otherwi


# Calculate mean of the numeric column
mean_value = df1.agg(mean(col("Electric Range"))).collect()[0][0]

# Replace null values with mean
imputed_df = imputed_df.withColumn("Electric Range", when(col("Electric Range").isNull(), mean_value).otherwise(col("Elec


# Show the DataFrame with imputed values
display(imputed_df)
```

Table

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make
1	5UXKT0C59G	Yakima	Zillah	WA	98953	2016	BMW
2	5YJ3E1EA2J	Snohomish	Edmonds	WA	98020	2018	TESLA
3	1G1RE6E4XE	Kitsap	Port Orchard	WA	98367	2014	CHEVROLET
4	2C4RC1L76M	Skagit	Bow	WA	98232	2021	CHRYSLER
5	5YJ3E1EA2J	Thurston	Olympia	WA	98513	2018	TESLA
6	WA1E2BFY8N	Snohomish	Snohomish	WA	98296	2022	AUDI

7,404 rows | Truncated data

```
#count number of null values in each column of DataFrame
imputed_df.select([count(when(col(c).isNull(), c)).alias(c) for c in imputed_df.columns]).show()
```

```
+----------+------+----+-----+-----------+----------+----+-----+--------------------+-----------------------------------
-------------+-------------+---------+-------------------+-------------+----------------+----------------+-----------
------+
|VIN (1-10)|County|City|State|Postal Code|Model Year|Make|Model|Electric Vehicle Type|Clean Alternative Fuel Vehicle (CAF
V) Eligibility|Electric Range|Base MSRP|Legislative District|DOL Vehicle ID|Vehicle Location|Electric Utility|2020 Census
Tract|
+----------+------+----+-----+-----------+----------+----+-----+--------------------+-----------------------------------
-------------+-------------+---------+-------------------+-------------+----------------+----------------+-----------
------+
|         0|     0|   0|    0|          0|         0|   0|    0|                   0|
0|            0|        1|                  0|            0|              10|               5|                5|
+----------+------+----+-----+-----------+----------+----+-----+--------------------+-----------------------------------
-------------+-------------+---------+-------------------+-------------+----------------+----------------+-----------
------+
```

```
# Count distinct values in the Vehicle Location column
len(imputed_df.select('Vehicle Location').distinct().collect())
```

```
Out[31]: 847
```

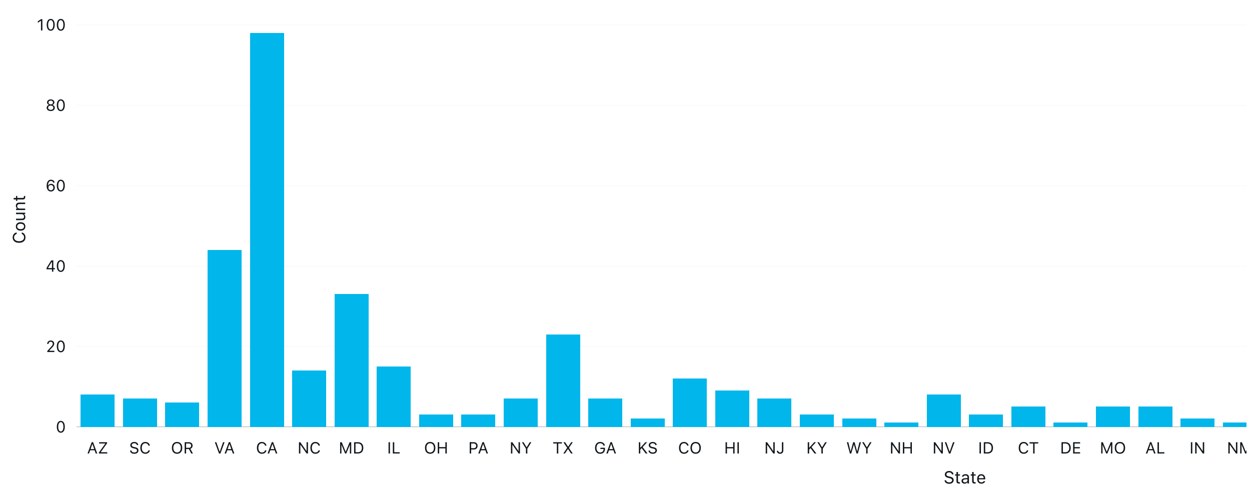
```
# Filter  rows where 'Vehicle Location' or 'Base MSRP' is null

imputed_df = imputed_df.filter(imputed_df['Vehicle Location'].isNotNull())
imputed_df = imputed_df.filter(imputed_df['Base MSRP'].isNotNull())
```

```
#check for null values in all columns after filtering

imputed_df.select([count(when(col(c).isNull(), c)).alias(c) for c in imputed_df.columns]).show()
```

```
+----------+------+----+-----+-----------+----------+----+-----+----------------------+------------------------------------
--------------+--------------+---------+--------------------+--------------+-----------------+----------------+-----------
------+
|VIN (1-10)|County|City|State|Postal Code|Model Year|Make|Model|Electric Vehicle Type|Clean Alternative Fuel Vehicle (CAF
V) Eligibility|Electric Range|Base MSRP|Legislative District|DOL Vehicle ID|Vehicle Location|Electric Utility|2020 Census
Tract|
+----------+------+----+-----+-----------+----------+----+-----+----------------------+------------------------------------
--------------+--------------+---------+--------------------+--------------+-----------------+----------------+-----------
------+
|         0|     0|   0|    0|          0|         0|   0|    0|                     0|
0|             0|        0|                   0|             0|                0|               0|                0|
+----------+------+----+-----+-----------+----------+----+-----+----------------------+------------------------------------
--------------+--------------+---------+--------------------+--------------+-----------------+----------------+-----------
------+
```

```
# imputed_df.(df.State < 40).count()
df_without_WA = imputed_df.filter(imputed_df.State != 'WA')
display(df_without_WA.groupBy('State').count())
```

Table | **Visualization 1**



42 rows

```
#filter the dataFrame to include  records from the WA state.

df_with_WA = imputed_df.filter(imputed_df.State == 'WA')
display(df_with_WA)
```

Table

	VIN (1-10)	County	City	State	Postal Code	Model Year	Make
1	5UXKT0C59G	Yakima	Zillah	WA	98953	2016	BMW
2	5YJ3E1EA2J	Snohomish	Edmonds	WA	98020	2018	TESLA
3	1G1RE6E4XE	Kitsap	Port Orchard	WA	98367	2014	CHEVROLET
4	2C4RC1L76M	Skagit	Bow	WA	98232	2021	CHRYSLER
5	5YJ3E1EA2J	Thurston	Olympia	WA	98513	2018	TESLA
6	WA1E2BFY8N	Snohomish	Snohomish	WA	98296	2022	AUDI

7,404 rows | Truncated data

```
#count the number of records for each city.

display(df_with_WA.groupBy('City').count())
```

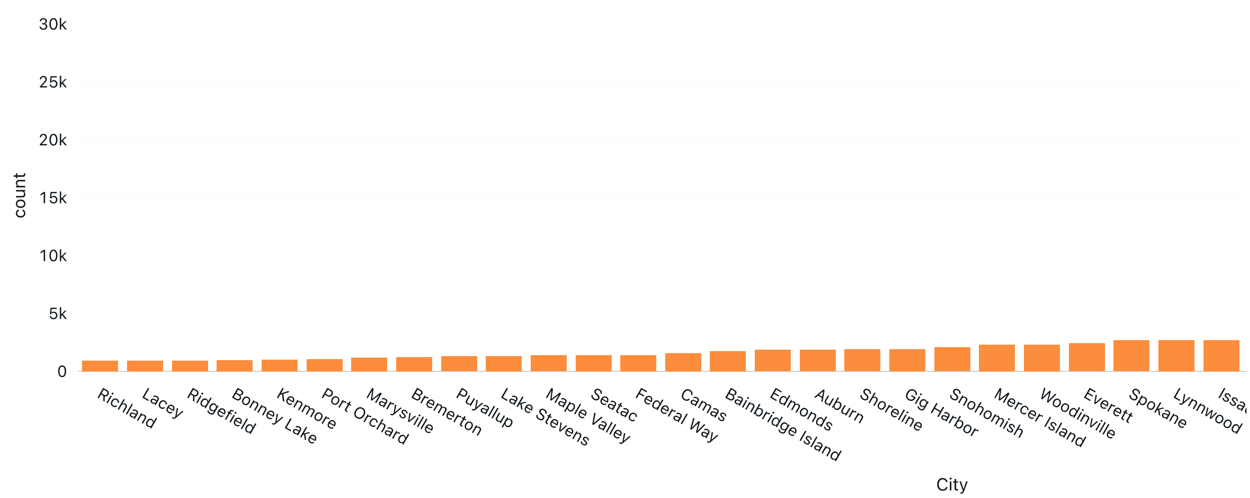
Table

	City	count
1	Bingen	7
2	Edmonds	1850
3	Bow	94
4	Pasco	537
5	Tumwater	618
6	Auburn	1854

464 rows

```
from pyspark.sql import functions as F

#group by city and count the number of records for each city and filter cities which have mor ethan 900 records.
df_WA_city_count = df_with_WA.groupBy('City').count()
df_WA_city_count_more900 = df_WA_city_count.filter(F.col('count') > 900)
display(df_WA_city_count_more900.orderBy('count'))
```

Table Visualization 1

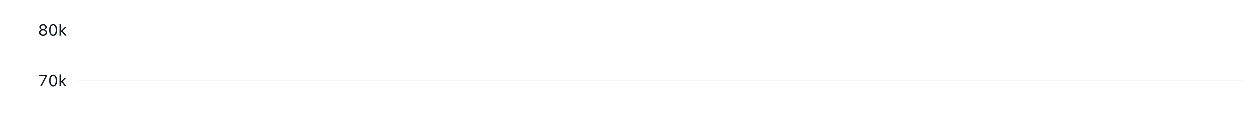


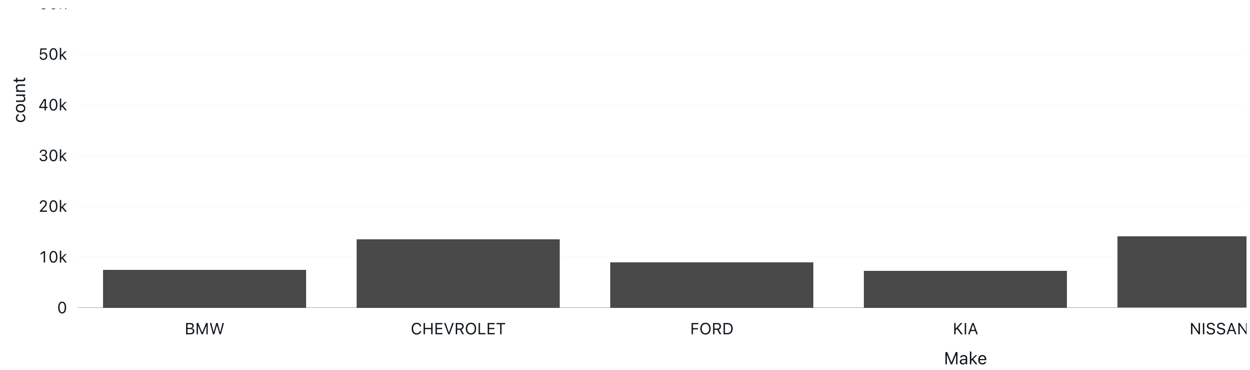
39 rows

```
brand_w_count = imputed_df.groupBy('Make').count()
brand_with_more_customers = brand_w_count.filter(F.col('count') > 5000)
brand_with_less_customers = brand_w_count.filter((F.col('count') < 5000) & (F.col('count') > 1000))
```

```
display(brand_with_more_customers)
```

Table Visualization 1

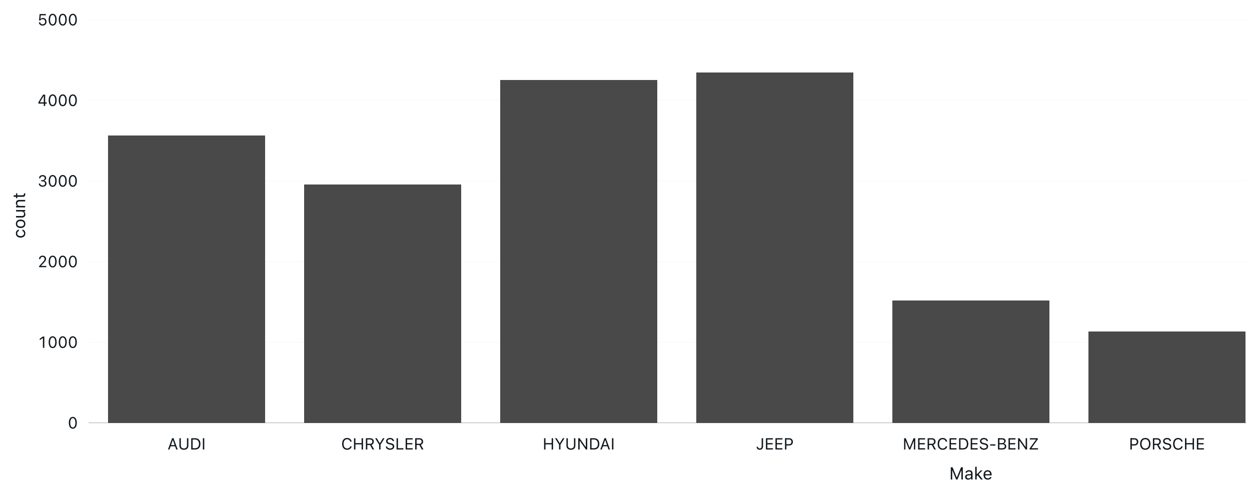




7 rows

```
display(brand_with_less_customers)
```

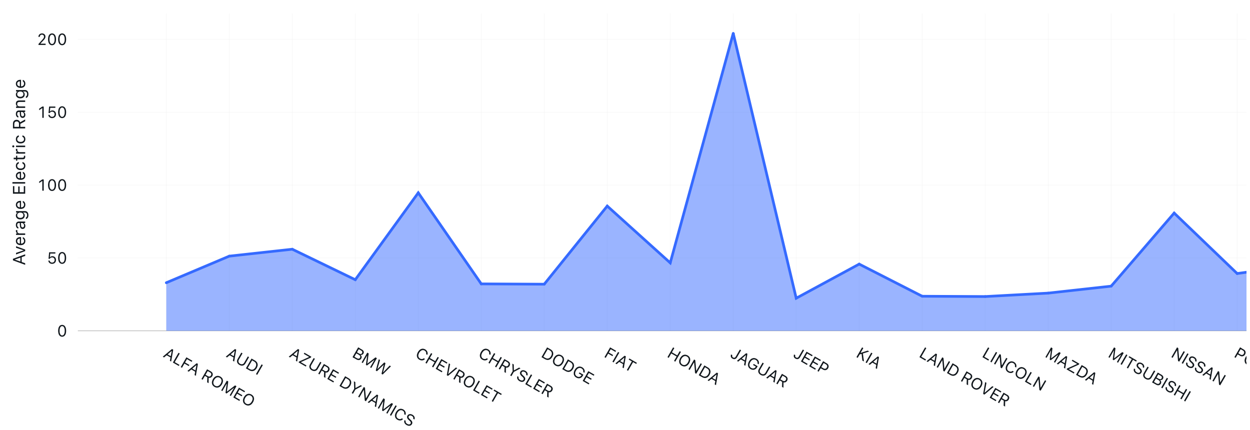
Table **Visualization 1**



9 rows

```
df_brand_range = imputed_df.groupBy('Make').agg({'Electric Range': 'mean'})
df_brand_range = df_brand_range.filter(F.col('avg(Electric Range)') > 20)
display(df_brand_range)
```

Table **Visualization 1**

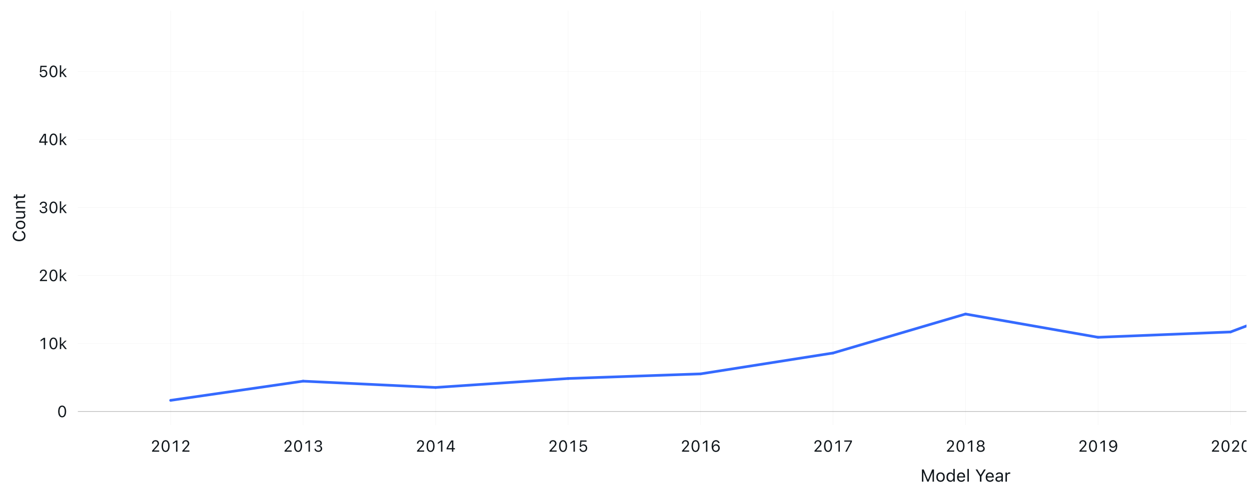


Make

25 rows

```
year_count = imputed_df.groupBy('Model Year').count()
year_count = year_count.filter(F.col('count')>1000)
display(year_count)
```

Table | Visualization 1



13 rows

```
#list numeric columns and select it from dataframe
numeric_column = ['Electric Range','Base MSRP','Legislative District']
numeric_column_df = df_with_WA.select(numeric_column)
display(numeric_column_df)
```

Table

	Electric Range	Base MSRP	Legislative District
1	14	0	15
2	215	0	21
3	38	0	26
4	32	0	40
5	215	0	2
6	23	0	1

10,000 rows | Truncated data

```
#Convert numeric column to int.
for column in numeric_column:
    df_with_WA = df_with_WA.withColumn(column, col(column).cast('int'))

#Split dataframe into test and train data
trainDF, testDF = df_with_WA.randomSplit([0.8, 0.2], seed=42)

print(trainDF.cache().count())
print(testDF.count())
```

```
138714
34437
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
from pyspark.ml.feature import StringIndexer, OneHotEncoder
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.feature import VectorAssembler

# Define categorical column
categoricalCols = [ 'County', 'City', 'State', 'Make', 'Model', 'Electric Vehicle Type', 'Vehicle Location', 'Electric Ut
categoricalCols.remove('State')
# Initialize StringIndexer for categorical features
stringIndexer = StringIndexer(inputCols=categoricalCols, outputCols=[x + "Index" for x in categoricalCols],handleInvalid=

# Initialize OneHotEncoder for the output of StringIndexer
encoder = OneHotEncoder(inputCols=stringIndexer.getOutputCols(), outputCols=[x + "OHE" for x in categoricalCols])

# Initialize StringIndexer for the label/target column
labelToIndex = StringIndexer(inputCol="Clean Alternative Fuel Vehicle (CAFV) Eligibility", outputCol="label")

# Combine all feature columns into a single feature vector
assemblerInputs = [c + "OHE" for c in categoricalCols] + numeric_column  # Note: using OHE columns
vecAssembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
```

```
from pyspark.ml import Pipeline
# Define the pipeline based on the stages created in previous steps.
pipeline = Pipeline(stages=[stringIndexer,encoder,vecAssembler,labelToIndex])

# Define the pipeline model.
pipelineModel = pipeline.fit(trainDF)
predDF = pipelineModel.transform(trainDF)
predDF1 = pipelineModel.transform(testDF)
# Define RandomForestClassifier
from pyspark.ml.classification import RandomForestClassifier
# Initialize the RandomForestClassifier
rfClassifier = RandomForestClassifier(featuresCol="features", labelCol="label", numTrees=10)

# Train the model on the transformed DataFrame
rfModel = rfClassifier.fit(predDF)

# Make predictions on the training data (you can also use a separate test set)
predictions = rfModel.transform(predDF1)
```

```
predictions.select("features", "label", "prediction", "probability").show()
```

```
+--------------------+-----+----------+--------------------+
|            features|label|prediction|         probability|
+--------------------+-----+----------+--------------------+
|(1290,[0,50,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[0,44,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[0,39,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[0,50,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[0,65,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[0,40,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[0,50,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[0,39,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[0,50,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[0,50,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[0,50,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[1,43,506,5...|  2.0|       2.0|[0.21229847726617...|
|(1290,[3,42,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[3,101,506,...|  2.0|       2.0|[0.20689909026626...|
|(1290,[16,159,506...|  2.0|       2.0|[0.20689909026626...|
```

```
|(1290,[0,40,506,5...|  2.0|       2.0|[0.20689909026626...|
|(1290,[0,50,506,5...|  2.0|       2.0|[0.20689909026626...|
```

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Calculate the accuracy of the predictions

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)

print("Accuracy:", accuracy)
```

```
Accuracy: 0.9006010976565902
```

```
#Extract the feature columns from the DataFrame and retrieve  feature importances.

feature_columns = predDF.columns[:-1]
feature_importances = rfModel.featureImportances

# Create a dictionary of feature importances
importances_dict = {feature_columns[i]: feature_importances[i] for i in range(len(feature_columns))}

# Print feature importances
for feature, importance in importances_dict.items():
    if importance >0:
        print(f"Feature: {feature}, Importance: {importance}")
```

```
Feature: Vehicle LocationIndex, Importance: 4.08680900413647e-05
Feature: Electric UtilityIndex, Importance: 3.5202241936741704e-05
Feature: ModelOHE, Importance: 3.818257512498391e-05
```

```
#reserach question 2(Which city consistently has electric cars that can drive the farthest on a single charge, considerin
from pyspark.sql.functions import col, min,max ,avg
from pyspark.sql.functions import desc, asc

# Average of electric range of each car model in each city.
city_avg_range = df_with_WA.groupBy("City", "Model Year", "Make", "Model","State").agg(avg("Electric Range").alias("Avg C
Electric Range In City"))

#Average of each car model.
car_avg_range = df_with_WA.groupBy("Model Year", "Make", "Model").agg(avg("Electric Range").alias("Avg Car Electric Range

# Perform an inner join between the DataFrames city_avg_range and car_avg_range.

inner_join_df = city_avg_range.join(car_avg_range, ["Model Year", "Make", "Model"], "inner")
display(inner_join_df.orderBy(desc("Avg Car Electric Range In City")))
```

Table

	Model Year	Make	Model	City	State	Avg Car Ele
1	2020	TESLA	MODEL S	Newcastle	WA	337
2	2020	TESLA	MODEL S	Tukwila	WA	337
3	2020	TESLA	MODEL S	Battle Ground	WA	337
4	2020	TESLA	MODEL S	Black Diamond	WA	337
5	2020	TESLA	MODEL S	Chattaroy	WA	337
6	2020	TESLA	MODEL S	Woodland	WA	337

10,000 rows | Truncated data

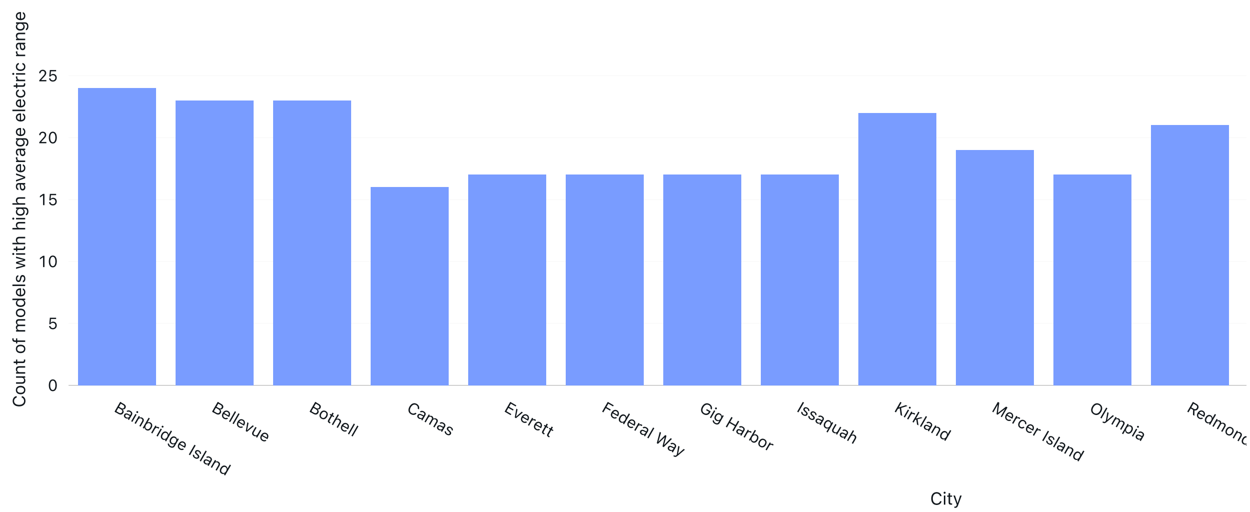
```
# Filter DataFrame to include only records where the average electric range in the city is greater than the overall avera

filter_range = inner_join_df.filter(inner_join_df["Avg Car Electric Range In City"] > inner_join_df["Avg Car Electric Ran
display(filter_range.orderBy(desc("Avg Car Electric Range In City")))
```

Table

	Model Year	Make	Model	City	State	Avg Car Electric Range In City
1	2020	TESLA	MODEL S	Battle Ground	WA	337
2	2020	TESLA	MODEL S	Newcastle	WA	337
3	2020	TESLA	MODEL S	Tukwila	WA	337
4	2020	TESLA	MODEL S	Chattaroy	WA	337
5	2020	TESLA	MODEL S	Woodland	WA	337
6	2020	TESLA	MODEL S	Carnation	WA	337

1,427 rows

Table Visualization 1



18 rows