Goals of US Accidents dataset analysis

- Understanding factors that contribute to traffic accidents
- Building a predictive model for accident severity
- Identifying high-risk locations or conditions
- Creating visualizations of accident patterns = Providing insights that could help reduce accidents

Setting Up PySpark

In [1]:
```
# # # !apt-get install openjdk-8-jdk-headless -qq > /dev/null
# !wget -q https://dlcdn.apache.org/spark/spark-3.5.5/spark-3.5.5-bin-hadoop3.tgz
# !tar xf spark-3.5.5-bin-hadoop3.tgz
# !pip install -q findspark
```

Libraries

In [2]:
```
import findspark
findspark.init()
from pyspark.sql import SparkSession

from pyspark.sql import functions as F
from pyspark.sql.types import *
from pyspark.ml.feature import Imputer, StringIndexer, OneHotEncoder, VectorAssembl
```

Loading Data

In [3]:
```
spark = SparkSession.builder \
    .appName("US Accidents Analysis") \
    .getOrCreate()
```

```
25/04/10 11:55:06 WARN Utils: Your hostname, lenovo-server resolves to a loopback ad
dress: 127.0.1.1; using 192.168.100.30 instead (on interface eno1)
25/04/10 11:55:06 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another addr
ess
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(ne
wLevel).
25/04/10 11:55:06 WARN NativeCodeLoader: Unable to load native-hadoop library for yo
ur platform... using builtin-java classes where applicable
```

In [4]:
```
df = spark.read.format("csv") \
    .option("header", "True") \
    .option("inferSchema", "True") \
    .load("US_Accidents_March23.csv")
```

```
25/04/10 11:55:17 WARN GarbageCollectionMetrics: To enable non-built-in garbage coll
ector(s) List(G1 Concurrent GC), users should configure it(them) to spark.eventLog.g
cMetrics.youngGenerationGarbageCollectors or spark.eventLog.gcMetrics.oldGenerationG
arbageCollectors
```

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

```
root
 |-- ID: string (nullable = true)
 |-- Source: string (nullable = true)
 |-- Severity: integer (nullable = true)
 |-- Start_Time: timestamp (nullable = true)
 |-- End_Time: timestamp (nullable = true)
 |-- Start_Lat: double (nullable = true)
 |-- Start_Lng: double (nullable = true)
 |-- End_Lat: double (nullable = true)
 |-- End_Lng: double (nullable = true)
 |-- Distance(mi): double (nullable = true)
 |-- Description: string (nullable = true)
 |-- Street: string (nullable = true)
 |-- City: string (nullable = true)
 |-- County: string (nullable = true)
 |-- State: string (nullable = true)
 |-- Zipcode: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Timezone: string (nullable = true)
 |-- Airport_Code: string (nullable = true)
 |-- Weather_Timestamp: timestamp (nullable = true)
 |-- Temperature(F): double (nullable = true)
 |-- Wind_Chill(F): double (nullable = true)
 |-- Humidity(%): double (nullable = true)
 |-- Pressure(in): double (nullable = true)
 |-- Visibility(mi): double (nullable = true)
 |-- Wind_Direction: string (nullable = true)
 |-- Wind_Speed(mph): double (nullable = true)
 |-- Precipitation(in): double (nullable = true)
 |-- Weather_Condition: string (nullable = true)
 |-- Amenity: boolean (nullable = true)
 |-- Bump: boolean (nullable = true)
 |-- Crossing: boolean (nullable = true)
 |-- Give_Way: boolean (nullable = true)
 |-- Junction: boolean (nullable = true)
 |-- No_Exit: boolean (nullable = true)
 |-- Railway: boolean (nullable = true)
 |-- Roundabout: boolean (nullable = true)
 |-- Station: boolean (nullable = true)
 |-- Stop: boolean (nullable = true)
 |-- Traffic_Calming: boolean (nullable = true)
 |-- Traffic_Signal: boolean (nullable = true)
 |-- Turning_Loop: boolean (nullable = true)
 |-- Sunrise_Sunset: string (nullable = true)
 |-- Civil_Twilight: string (nullable = true)
 |-- Nautical_Twilight: string (nullable = true)
 |-- Astronomical_Twilight: string (nullable = true)
```

In [6]:
```
df.show(5)
```

```
25/04/10 11:55:22 WARN SparkStringUtils: Truncated the string representation of a pl
an since it was too large. This behavior can be adjusted by setting 'spark.sql.debu
g.maxToStringFields'.
```

```
+---+-------+--------+------------------+------------------+-----------------
+------------------+-------+-------+------------+--------------------
+--------------------+------------+----------+-----+----------+-------+----------
+------------+------------------+--------------+-------------+-----------
+------------+--------------+--------------+---------------+-----------------
+-----------------+-------+-----+--------+--------+--------+-------+-------
+----------+-------+-----+---------------+--------------+------------+--------------
+--------------+-----------------+---------------------+
| ID| Source|Severity|         Start_Time|           End_Time|        Start_Lat|
Start_Lng|End_Lat|End_Lng|Distance(mi)|         Description|              Street|
City|    County|State|   Zipcode|Country|  Timezone|Airport_Code|  Weather_Timestam
p|Temperature(F)|Wind_Chill(F)|Humidity(%)|Pressure(in)|Visibility(mi)|Wind_Directio
n|Wind_Speed(mph)|Precipitation(in)|Weather_Condition|Amenity| Bump|Crossing|Give_Wa
y|Junction|No_Exit|Railway|Roundabout|Station| Stop|Traffic_Calming|Traffic_Signal|T
urning_Loop|Sunrise_Sunset|Civil_Twilight|Nautical_Twilight|Astronomical_Twilight|
+---+-------+--------+------------------+------------------+-----------------
+------------------+-------+-------+------------+--------------------
+--------------------+------------+----------+-----+----------+-------+----------
+------------+------------------+--------------+-------------+-----------
+------------+--------------+--------------+---------------+-----------------
+-----------------+-------+-----+--------+--------+--------+-------+-------
+----------+-------+-----+---------------+--------------+------------+--------------
+--------------+-----------------+---------------------+
|A-1|Source2|       3|2016-02-08 05:46:00|2016-02-08 11:00:00|        39.865147|
-84.058723|   NULL|   NULL|        0.01|Right lane blocke...|              I-70 E|
Dayton|Montgomery|   OH|     45424|     US|US/Eastern|        KFFO|2016-02-08 05:5
8:00|          36.9|         NULL|       91.0|       29.68|          10.0|
Calm|           NULL|              0.02|       Light Rain|  false|false|   false|   f
alse|   false|  false|  false|     false|  false|false|          false|         fals
e|       false|         Night|         Night|           Night|               Nigh
t|
|A-2|Source2|       2|2016-02-08 06:07:59|2016-02-08 06:37:59|39.92805900000001|
-82.831184|   NULL|   NULL|        0.01|Accident on Brice...|            Brice Rd|Re
ynoldsburg|  Franklin|   OH|43068-3402|     US|US/Eastern|        KCMH|2016-02-08 0
5:51:00|          37.9|         NULL|      100.0|       29.65|          10.0|
Calm|           NULL|              0.0|       Light Rain|  false|false|   false|   f
alse|   false|  false|  false|     false|  false|false|          false|         fals
e|       false|         Night|         Night|           Night|                 Da
y|
|A-3|Source2|       2|2016-02-08 06:49:27|2016-02-08 07:19:27|        39.063148|
-84.032608|   NULL|   NULL|        0.01|Accident on OH-32...|      State Route 32|Wi
lliamsburg|  Clermont|   OH|     45176|     US|US/Eastern|        KI69|2016-02-08 0
6:56:00|          36.0|         33.3|      100.0|       29.67|          10.0|
SW|            3.5|             NULL|         Overcast|  false|false|   false|   fal
se|   false|  false|  false|     false|  false|false|          false|          true|
false|         Night|         Night|             Day|                 Day|
|A-4|Source2|       3|2016-02-08 07:23:34|2016-02-08 07:53:34|        39.747753|-8
4.20558199999998|   NULL|   NULL|        0.01|Accident on I-75 ...|              I-7
5 S|      Dayton|Montgomery|   OH|     45417|     US|US/Eastern|        KDAY|2016-0
2-08 07:38:00|          35.1|         31.0|       96.0|       29.64|           9.0|
SW|            4.6|             NULL|    Mostly Cloudy|  false|false|   false|   fal
se|   false|  false|  false|     false|  false|false|          false|         false|
false|         Night|           Day|             Day|                 Day|
|A-5|Source2|       2|2016-02-08 07:39:07|2016-02-08 08:09:07|        39.627781|
-84.188354|   NULL|   NULL|        0.01|Accident on McEwe...|Miamisburg Center...|
Dayton|Montgomery|   OH|     45459|     US|US/Eastern|        KMGY|2016-02-08 07:5
```

```
3:00|          36.0|         33.3|       89.0|       29.65|           6.0|
SW|            3.5|             NULL|    Mostly Cloudy|  false|false|   false|   fal
se|   false|  false|  false|     false|  false|false|          false|          true|
false|           Day|           Day|              Day|                  Day|
+---+-------+--------+------------------+------------------+-----------------
+------------------+-------+-------+-----------+-------------------
+--------------------+-----------+----------+-----+----------+-------+----------
+------------+------------------+--------------+-------------+-----------
+------------+--------------+--------------+---------------+-----------------
+-----------------+-------+-----+--------+--------+--------+-------+-------
+----------+-------+-----+--------------+-------------+------------+--------------
+--------------+-----------------+--------------------+
only showing top 5 rows
```

In [7]:
```
print(f"Total number of records: {df.count()}")
print(f"Number of columns: {len(df.columns)}")
print("Column names:", df.columns)
```

```
[Stage 3:=======================================>                 (16 + 6) / 23]
Total number of records: 7728394
Number of columns: 46
Column names: ['ID', 'Source', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'S
tart_Lng', 'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Street', 'City', 'C
ounty', 'State', 'Zipcode', 'Country', 'Timezone', 'Airport_Code', 'Weather_Timestam
p', 'Temperature(F)', 'Wind_Chill(F)', 'Humidity(%)', 'Pressure(in)', 'Visibility(m
i)', 'Wind_Direction', 'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition',
'Amenity', 'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway', 'Rounda
bout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal', 'Turning_Loop', 'Sunr
ise_Sunset', 'Civil_Twilight', 'Nautical_Twilight', 'Astronomical_Twilight']
```

Data preprocess and clean

In [8]:
```
print("Missing values per column:")
for column_name, dtype in df.dtypes:
    if dtype in ("int", "double", "float"):
        # For numeric columns, check for nulls and NaN values
        missing_count = df.filter(F.col(column_name).isNull() | F.isnan(column_name
    else:
        # For non-numeric columns, only check for nulls and empty strings
        missing_count = df.filter(F.col(column_name).isNull() | (F.col(column_name)

    print(f"{column_name}: {missing_count}")
```

```
Missing values per column:

ID: 0

Source: 0

Severity: 0
```

```
Start_Time: 0

End_Time: 0

Start_Lat: 0

Start_Lng: 0

End_Lat: 3402762

End_Lng: 3402762

Distance(mi): 0

Description: 5

Street: 10869

City: 253

County: 0

State: 0

Zipcode: 1915

Country: 0

Timezone: 7808

Airport_Code: 22635

Weather_Timestamp: 120228

Temperature(F): 163853

Wind_Chill(F): 1999019

Humidity(%): 174144

Pressure(in): 140679

Visibility(mi): 177098

Wind_Direction: 175206

Wind_Speed(mph): 571233

Precipitation(in): 2203586
```

```
Weather_Condition: 173459

Amenity: 0

Bump: 0

Crossing: 0

Give_Way: 0

Junction: 0

No_Exit: 0

Railway: 0

Roundabout: 0

Station: 0

Stop: 0

Traffic_Calming: 0

Traffic_Signal: 0

Turning_Loop: 0

Sunrise_Sunset: 23246

Civil_Twilight: 23246

Nautical_Twilight: 23246
[Stage 141:=============================================>           (18 + 5) / 23]
Astronomical_Twilight: 23246
```

Dealing with missing values

End_Lat and End_Lng Missing values

For End_Lat and End_Lng: Assuming, these are likely accidents where the endpoint wasn't recorded

Dropping these columns, fill with Start_Lat/Start_Lng, or keep nulls

In [13]:
```
df = df.withColumn("End_Lat", F.when(F.col("End_Lat").isNull(), F.col("Start_Lat"))
df = df.withColumn("End_Lng", F.when(F.col("End_Lng").isNull(), F.col("Start_Lng"))
```

Weather-related columns Missing Values

Numeric

We have options:

- Fill with mean/median (for numeric data)
- Fill with a special value indicating "unknown"
- Leave as null for downstream processing

In [14]:
```
weather_numeric_cols = ["Temperature(F)", "Wind_Chill(F)", "Humidity(%)",
                        "Pressure(in)", "Visibility(mi)", "Wind_Speed(mph)",
                        "Precipitation(in)"]

for col in weather_numeric_cols:
    # Calculate mean excluding null values
    mean_val = df.select(F.mean(col)).collect()[0][0]
    # Replace nulls with the mean
    df = df.withColumn(col, F.when(F.col(col).isNull(), mean_val).otherwise(F.col(c
```

Categorical

For categorical weather data, replacing missing with "Unknown"

In [15]:
```
df = df.withColumn("Weather_Condition",
                   F.when(F.col("Weather_Condition").isNull(), "Unknown")
                    .otherwise(F.col("Weather_Condition")))

df = df.withColumn("Wind_Direction",
                   F.when(F.col("Wind_Direction").isNull(), "Unknown")
                    .otherwise(F.col("Wind_Direction")))
```

Twilight-related columns Missing Values

Filling with most frequent value

In [16]:
```
twilight_cols = ["Sunrise_Sunset", "Civil_Twilight", "Nautical_Twilight", "Astronom
for col in twilight_cols:
    # Fill with most frequent value
    most_common = df.groupBy(col).count().orderBy("count", ascending=False).first()
    df = df.withColumn(col, F.when(F.col(col).isNull(), most_common).otherwise(F.co
```

Other location based and description

In [17]:
```
df = df.withColumn("Street", F.when(F.col("Street").isNull(), "Unknown").otherwise(
df = df.withColumn("City", F.when(F.col("City").isNull(), "Unknown").otherwise(F.co
df = df.withColumn("Zipcode", F.when(F.col("Zipcode").isNull(), "Unknown").otherwis
df = df.withColumn("Airport_Code", F.when(F.col("Airport_Code").isNull(), "Unknown"
df = df.withColumn("Timezone", F.when(F.col("Timezone").isNull(), "Unknown").otherw
df = df.withColumn("Description", F.when(F.col("Description").isNull(), "No descrip
```

Missing Weather_Timestamp with Start_Time

In [18]:
```
df = df.withColumn("Weather_Timestamp",
                   F.when(F.col("Weather_Timestamp").isNull(), F.col("Start_Time"))
                    .otherwise(F.col("Weather_Timestamp")))
```

After Clean

In [19]:
```
print("Missing values after cleaning:")
for column_name in df.columns:
    missing_count = df.filter(F.col(column_name).isNull()).count()
    print(f"{column_name}: {missing_count}")
```

```
Missing values after cleaning:

ID: 0

Source: 0

Severity: 0

Start_Time: 0

End_Time: 0

Start_Lat: 0

Start_Lng: 0

End_Lat: 0

End_Lng: 0

Distance(mi): 0

Description: 0

Street: 0
```

```
City: 0

County: 0

State: 0

Zipcode: 0

Country: 0

Timezone: 0

Airport_Code: 0

Weather_Timestamp: 0

Temperature(F): 0

Wind_Chill(F): 0

Humidity(%): 0

Pressure(in): 0

Visibility(mi): 0

Wind_Direction: 0

Wind_Speed(mph): 0

Precipitation(in): 0

Weather_Condition: 0

Amenity: 0

Bump: 0

Crossing: 0

Give_Way: 0

Junction: 0

No_Exit: 0

Railway: 0
```

```
Roundabout: 0

Station: 0

Stop: 0

Traffic_Calming: 0

Traffic_Signal: 0

Turning_Loop: 0

Sunrise_Sunset: 0

Civil_Twilight: 0

Nautical_Twilight: 0
[Stage 471:=============================================>           (18 + 5) / 23]
Astronomical_Twilight: 0
```

In [21]:
```
df.show(5)
```

```
+---+-------+--------+------------------+------------------+-----------------
+------------------+-----------------+------------------+------------
+--------------------+--------------------+------------+----------+-----+----------
+-------+----------+------------+------------------+--------------
+-----------------+-----------+------------+--------------+--------------
+-----------------+--------------------+-----------------+-------+-----+--------
+--------+--------+-------+-------+----------+-------+-----+---------------
+--------------+------------+--------------+--------------+-----------------
+---------------------+
| ID| Source|Severity|         Start_Time|           End_Time|        Start_Lat|
Start_Lng|          End_Lat|           End_Lng|Distance(mi)|         Description|
Street|        City|    County|State|   Zipcode|Country|  Timezone|Airport_Code|  We
ather_Timestamp|Temperature(F)|    Wind_Chill(F)|Humidity(%)|Pressure(in)|Visibility
(mi)|Wind_Direction|  Wind_Speed(mph)|   Precipitation(in)|Weather_Condition|Amenit
y| Bump|Crossing|Give_Way|Junction|No_Exit|Railway|Roundabout|Station| Stop|Traffic_
Calming|Traffic_Signal|Turning_Loop|Sunrise_Sunset|Civil_Twilight|Nautical_Twilight|
Astronomical_Twilight|
+---+-------+--------+------------------+------------------+-----------------
+------------------+-----------------+------------------+------------
+--------------------+--------------------+------------+----------+-----+----------
+-------+----------+------------+------------------+--------------
+-----------------+-----------+------------+--------------+--------------
+-----------------+--------------------+-----------------+-------+-----+--------
+--------+--------+-------+-------+----------+-------+-----+---------------
+--------------+------------+--------------+--------------+-----------------
+---------------------+
|A-1|Source2|       3|2016-02-08 05:46:00|2016-02-08 11:00:00|        39.865147|
-84.058723|        39.865147|        -84.058723|        0.01|Right lane blocke...|
I-70 E|      Dayton|Montgomery|   OH|     45424|     US|US/Eastern|        KFFO|201
6-02-08 05:58:00|          36.9|58.25104839533092|       91.0|       29.68|
10.0|          Calm|7.685489595665597|                0.02|       Light Rain|  fals
e|false|   false|   false|   false|  false|  false|     false|  false|false|
false|         false|       false|         Night|         Night|            Night|
Night|
|A-2|Source2|       2|2016-02-08 06:07:59|2016-02-08 06:37:59|39.92805900000001|
-82.831184|39.92805900000001|        -82.831184|        0.01|Accident on Brice...|
Brice Rd|Reynoldsburg|  Franklin|   OH|43068-3402|     US|US/Eastern|        KCMH|20
16-02-08 05:51:00|          37.9|58.25104839533092|      100.0|       29.65|
10.0|          Calm|7.685489595665597|                 0.0|       Light Rain|  fals
e|false|   false|   false|   false|  false|  false|     false|  false|false|
false|         false|       false|         Night|         Night|            Night|
Day|
|A-3|Source2|       2|2016-02-08 06:49:27|2016-02-08 07:19:27|        39.063148|
-84.032608|        39.063148|        -84.032608|        0.01|Accident on OH-32...|
State Route 32|Williamsburg|  Clermont|   OH|     45176|     US|US/Eastern|        K
I69|2016-02-08 06:56:00|          36.0|             33.3|      100.0|       29.67|
10.0|            SW|             3.5|0.008407209807109432|         Overcast|  fals
e|false|   false|   false|   false|  false|  false|     false|  false|false|
false|          true|       false|         Night|         Night|              Day|
Day|
|A-4|Source2|       3|2016-02-08 07:23:34|2016-02-08 07:53:34|        39.747753|-8
4.20558199999998|        39.747753|-84.20558199999998|        0.01|Accident on I-75
...|             I-75 S|      Dayton|Montgomery|   OH|     45417|     US|US|Easter
n|        KDAY|2016-02-08 07:38:00|          35.1|             31.0|       96.0|
29.64|           9.0|            SW|             4.6|0.008407209807109432|    Mostl
y Cloudy|  false|false|   false|   false|   false|  false|  false|     false|  fals
```

```
e|false|          false|         false|       false|         Night|           Day|
Day|                 Day|
|A-5|Source2|       2|2016-02-08 07:39:07|2016-02-08 08:09:07|        39.627781|
-84.188354|        39.627781|        -84.188354|        0.01|Accident on McEwe...|Mi
amisburg Center...|      Dayton|Montgomery|   OH|     45459|     US|US/Eastern|
KMGY|2016-02-08 07:53:00|          36.0|             33.3|       89.0|       29.65|
6.0|            SW|              3.5|0.008407209807109432|    Mostly Cloudy|  false|
false|   false|   false|   false|  false|  false|     false|  false|false|
false|          true|       false|           Day|           Day|              Day|
Day|
+---+-------+--------+------------------+------------------+-----------------
+------------------+-----------------+------------------+-----------
+--------------------+--------------------+-----------+----------+-----+----------
+-------+----------+-----------+------------------+-------------
+-----------------+----------+-----------+-------------+-------------
+-----------------+--------------------+-----------------+-------+-----+--------
+--------+--------+-------+-------+----------+-------+-----+---------------
+-------------+-----------+-------------+-------------+-----------------
+---------------------+
only showing top 5 rows
```

Data Preprocessing

In [27]:
```
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler, Stand
```

This classification is crucial because each data type requires different preprocessing approaches.

In [28]:
```
categorical_cols = [
    "Source", "Severity", "State", "Country", "Timezone", "Weather_Condition",
    "Wind_Direction", "Sunrise_Sunset", "Civil_Twilight", "Nautical_Twilight",
    "Astronomical_Twilight"
]

boolean_cols = [
    "Amenity", "Bump", "Crossing", "Give_Way", "Junction", "No_Exit",
    "Railway", "Roundabout", "Station", "Stop", "Traffic_Calming",
    "Traffic_Signal", "Turning_Loop"
]

numerical_cols = [
    "Start_Lat", "Start_Lng", "End_Lat", "End_Lng", "Distance(mi)",
    "Temperature(F)", "Wind_Chill(F)", "Humidity(%)", "Pressure(in)",
    "Visibility(mi)", "Wind_Speed(mph)", "Precipitation(in)"
]
```

Step 1: String Indexing

This creates a collection of StringIndexer transformers that:

- Convert each text value in categorical columns to a numerical index

- Assign indices based on frequency (most frequent value gets index 0)
- Create new columns with suffix "_idx" containing these numerical indices
- Handle invalid or unseen values gracefully with the "keep" option

In [29]:
```
indexers = [StringIndexer(inputCol=col, outputCol=f"{col}_idx", handleInvalid="keep
            for col in categorical_cols]
```

Step 2: One-Hot Encoding

This step transforms the indexed values into one-hot encoded vectors:

- Each category becomes a sparse vector (mostly zeros with a single 1)
- The vector length equals the number of unique values in the category
- The resulting columns have suffix "_ohe" (one-hot encoded)
- This transformation is essential as most ML algorithms cannot directly process categorical data

In [30]:
```
encoders = [OneHotEncoder(inputCol=f"{col}_idx", outputCol=f"{col}_ohe")
            for col in categorical_cols]
```

Feature Vector Creation

Step 3: Feature Column Collection

This collects all processed feature column names that will be used to build the final feature vector:

- One-hot encoded categorical columns
- Original numerical columns
- Original boolean columns (already in 0/1 format)

In [31]:
```
feature_cols = [f"{col}_ohe" for col in categorical_cols] + numerical_cols + boolea
```

Step 4: Vector Assembly

The VectorAssembler:

- Takes all individual feature columns
- Combines them into a single dense or sparse vector
- Creates a new column called "features" containing these vectors
- Properly handles any invalid values

In [32]:
```
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features", handleInv
```

Feature Standardization

This transformer standardizes the assembled feature vectors by:

- Subtracting the mean from each feature
- Dividing by the standard deviation
- Producing a new column "scaled_features" with standardized values
- Ensuring all features contribute equally to model learning regardless of their original scale

In [33]:
```
scaler = StandardScaler(inputCol="features", outputCol="scaled_features")
```

Pipeline Construction

The Pipeline combines all preprocessing steps into a single workflow:

1. First apply all string indexers to convert categories to numbers
2. Then apply all one-hot encoders to convert numbers to vectors
3. Assemble all features into a single vector
4. Scale the assembled vector

This ensures transformations are applied in the correct order and simplifies deployment.

In [34]:
```
from pyspark.ml import Pipeline

preprocessing_pipeline = Pipeline(stages=indexers + encoders + [assembler, scaler])
preprocessed_data = preprocessing_pipeline.fit(df).transform(df)
preprocessed_data.select("features", "scaled_features").show(5, truncate=True)
```

```
+--------------------+--------------------+
|            features|     scaled_features|
+--------------------+--------------------+
|(265,[1,4,25,56,5...|(265,[1,4,25,56,5...|
|(265,[1,3,25,56,5...|(265,[1,3,25,56,5...|
|(265,[1,3,25,56,5...|(265,[1,3,25,56,5...|
|(265,[1,4,25,56,5...|(265,[1,4,25,56,5...|
|(265,[1,3,25,56,5...|(265,[1,3,25,56,5...|
+--------------------+--------------------+
only showing top 5 rows
```