Report

Vehicle Insurance Fraud Detection

Introduction to Cluster Computing

Name: Mansi

Reg no.: 12222258

Roll no. : 65

Section: K22KW

Submitted to: Richa Mam

Introduction

Insurance fraud is one of the major challenges faced by the automobile insurance industry. Detecting fraudulent claims early can help reduce financial losses and improve operational efficiency.
The main goal of this project is to build a machine learning model using PySpark that can classify whether a given insurance claim is fraudulent or genuine, based on the characteristics of the claim and the policyholder.

This project uses a Decision Tree Classifier to predict fraudulent claims and evaluates the model's performance using Precision and Recall metrics.

Dataset Details

- **Dataset Name:** Car Insurance Claims (Vehicle Insurance Fraud Detection)
- **Source:** Kaggle - Vehicle Insurance Fraud Detection Dataset
- **File Name:** carclaims.csv
- **File Path (Local):** D:\7th SEM\INT 315\CA2\carclaims.csv

Data Summary

Total Rows: 15,420

Total Columns: 33

Numerical Columns: 8(e.g., Age, Deductible, DriverRating, etc.)

Categorical Columns: 25 (e.g., Make, AccidentArea, PolicyType, etc.)

Missing Values: None detected

The dataset was analysed using PySpark to understand its structure, size, and attribute distribution. The following table summarizes the key characteristics of the dataset:

```
>>> rows = df.count()
>>> cols = len(df.columns)
>>> print(f"Total Rows: {rows}")
Total Rows: 15420
>>> print(f"Total Columns: {cols}")
Total Columns: 33
>>> from pyspark.sql.functions import col, sum
>>>
>>> missing_df = df.select([sum(col(c).isNull().cast("int")).alias(c) for c in df.columns])
>>> missing_df.show()
+-----+----------+---------+----+-----------+---------------+-----------+----------------
-----+---------+----------+-----------+-------------------+----------------+--------------
+-------------------+------------------+------------+----+----------+----------+
|Month|WeekOfMonth|DayOfWeek|Make|AccidentArea|DayOfWeekClaimed|MonthClaimed|WeekOfMonthClaime
umber|RepNumber|Deductible|DriverRating|Days:Policy-Accident|Days:Policy-Claim|PastNumberOfCla
|NumberOfSuppliments|AddressChange-Claim|NumberOfCars|Year|BasePolicy|FraudFound|
+-----+----------+---------+----+-----------+---------------+-----------+----------------
-----+---------+----------+-----------+-------------------+----------------+--------------
+-------------------+------------------+------------+----+----------+----------+
```

```
>>> num_cols = [c for c, t in df.dtypes if t in ['int', 'double', 'float']]
>>> cat_cols = [c for c, t in df.dtypes if t == 'string']
>>> print(f"Numerical Columns: {len(num_cols)}")
Numerical Columns: 8
>>> print(f"Categorical Columns: {len(cat_cols)}")
Categorical Columns: 25
>>> print("\nNumerical Columns:", num_cols)

Numerical Columns: ['WeekOfMonth', 'WeekOfMonthClaimed', 'Age', 'PolicyNumber', 'RepNumber', 'Deducti
>>> print("\nCategorical Columns:", cat_cols)

Categorical Columns: ['Month', 'DayOfWeek', 'Make', 'AccidentArea', 'DayOfWeekClaimed', 'MonthClaimed
VehiclePrice', 'Days:Policy-Accident', 'Days:Policy-Claim', 'PastNumberOfClaims', 'AgeOfVehicle', 'Ag
'NumberOfSuppliments', 'AddressChange-Claim', 'NumberOfCars', 'BasePolicy', 'FraudFound']
>>>
```

Column Name	Description
Month	Month when the policy was issued
WeekOfMonth	Week of the month
DayOfWeek	Day of the week when the accident occurred
Make	Car manufacturer
AccidentArea	Area where accident happened (Urban/Rural)
Sex	Gender of policyholder
MaritalStatus	Marital status of policyholder
Age	Age of policyholder
VehicleCategory	Type of vehicle

Column Name	Description
VehiclePrice	Price range of the vehicle
Fault	Party at fault
PolicyType	Type of insurance policy
AgeOfVehicle	Age of the vehicle
PoliceReportFiled	Whether police report was filed
WitnessPresent	Whether a witness was present
BasePolicy	Type of base policy
FraudFound	Target column (1 = Fraudulent, 0 = Genuine)

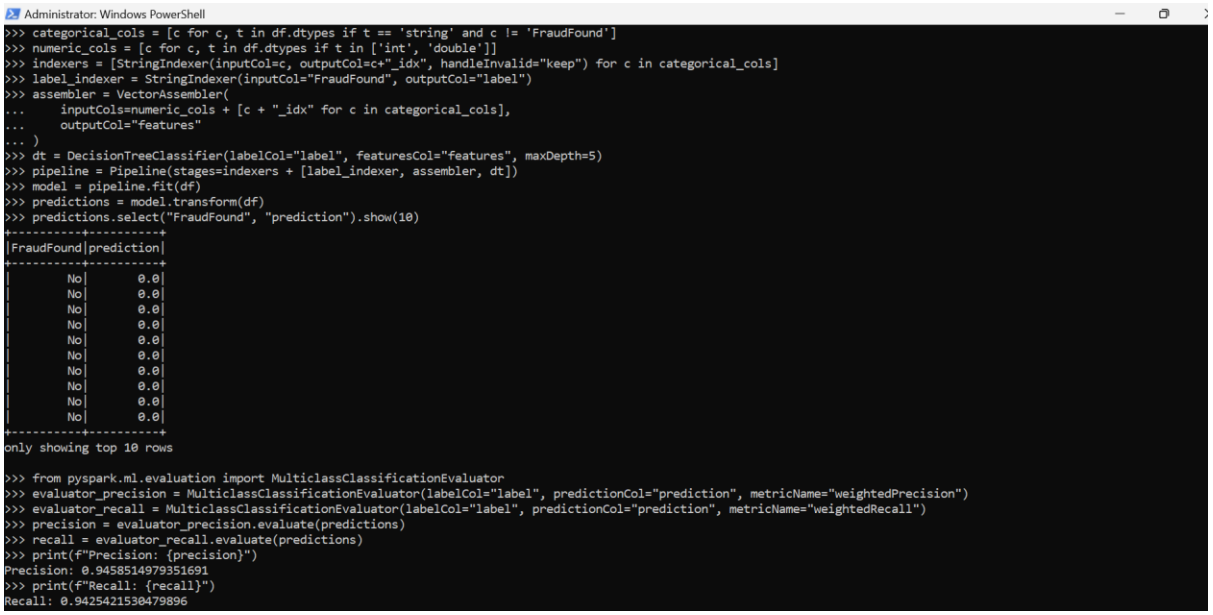
Source Code

The machine learning workflow was implemented in PySpark. The following steps outline the major stages of the implementation process:

Importing libraries, creating spark session, load dataset

```
>>> from pyspark.sql import SparkSession
>>> from pyspark.ml.feature import StringIndexer, VectorAssembler
>>> from pyspark.ml.classification import DecisionTreeClassifier
>>> from pyspark.ml.evaluation import MulticlassClassificationEvaluator
>>> from pyspark.ml import Pipeline
>>>
>>> spark = SparkSession.builder.appName("Vehicle_Insurance_Fraud_Detection").getOrCreate()
25/11/08 16:48:16 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>>
>>> df = spark.read.csv(r"D:\7th SEM\INT 315\CA2\carclaims.csv", header=True, inferSchema=True)
>>> df.show(5)
25/11/08 16:48:23 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.max
ToStringFields'.
+-----+-----------+---------+------+------------+----------------+------------+-----------------+------+-------------+---+------------+----------------+----------------+-
-------------+-----------+---------+----------+------------+-----------------+----------------+------------------+------------+-----------------+------------------+--
-----------+---------+-------------------+-------------------+------------+----+----------+----------+
|Month|WeekOfMonth|DayOfWeek|  Make|AccidentArea|DayOfWeekClaimed|MonthClaimed|WeekOfMonthClaimed|   Sex|MaritalStatus|Age|        Fault|       PolicyType|VehicleCategory|
   VehiclePrice|PolicyNumber|RepNumber|Deductible|DriverRating|Days:Policy-Accident|Days:Policy-Claim|PastNumberOfClaims|AgeOfVehicle|AgeOfPolicyHolder|PoliceReportFiled|Wi
tnessPresent|AgentType|NumberOfSuppliments|AddressChange-Claim|NumberOfCars|Year|BasePolicy|FraudFound|
+-----+-----------+---------+------+------------+----------------+------------+-----------------+------+-------------+---+------------+----------------+----------------+-
-------------+-----------+---------+----------+------------+-----------------+----------------+------------------+------------+-----------------+------------------+--
-----------+---------+-------------------+-------------------+------------+----+----------+----------+
|  Dec|          5|Wednesday| Honda|       Urban|         Tuesday|         Jan|                1|Female|       Single| 21|Policy Holder|Sport - Liability|          Sport|m
ore than 69,000|           1|       12|       300|           1|     more than 30|    more than 30|              none|     3 years|         26 to 30|                No|
         No| External|               none|            1 year|     3 to 4|1994| Liability|         No|
|  Jan|          3|Wednesday| Honda|       Urban|          Monday|         Jan|                4|  Male|       Single| 34|Policy Holder|Sport - Collision|          Sport|m
ore than 69,000|           2|       15|       400|           4|     more than 30|    more than 30|              none|     6 years|         31 to 35|               Yes|
         No| External|               none|         no change|  1 vehicle|1994| Collision|         No|
|  Oct|          5|   Friday| Honda|       Urban|        Thursday|         Nov|                2|  Male|      Married| 47|Policy Holder|Sport - Collision|          Sport|m
ore than 69,000|           3|        7|       400|           3|     more than 30|    more than 30|                 1|     7 years|         41 to 50|                No|
         No| External|               none|         no change|  1 vehicle|1994| Collision|         No|
|  Jun|          2| Saturday|Toyota|       Rural|          Friday|         Jul|                1|  Male|      Married| 65|  Third Party|Sedan - Liability|          Sport|2
0,000 to 29,000|           4|        4|       400|           2|     more than 30|    more than 30|                 1| more than 7|         51 to 65|               Yes|
         No| External|         more than 5|         no change|  1 vehicle|1994| Liability|         No|
|  Jan|          5|   Monday| Honda|       Urban|         Tuesday|         Feb|                2|Female|       Single| 27|  Third Party|Sport - Collision|          Sport|m
ore than 69,000|           5|        3|       400|           1|     more than 30|    more than 30|              none|     5 years|         31 to 35|                No|
         No| External|               none|         no change|  1 vehicle|1994| Collision|         No|
+-----+-----------+---------+------+------------+----------------+------------+-----------------+------+-------------+---+------------+----------------+----------------+-
-------------+-----------+---------+----------+------------+-----------------+----------------+------------------+------------+-----------------+------------------+--
-----------+---------+-------------------+-------------------+------------+----+----------+----------+
only showing top 5 rows
```

Encode Categorical Columns, Feature Vector Decision Tree Classifier, Pipeline , Train model



```
Administrator: Windows PowerShell
>>> categorical_cols = [c for c, t in df.dtypes if t == 'string' and c != 'FraudFound']
>>> numeric_cols = [c for c, t in df.dtypes if t in ['int', 'double']]
>>> indexers = [StringIndexer(inputCol=c, outputCol=c+"_idx", handleInvalid="keep") for c in categorical_cols]
>>> label_indexer = StringIndexer(inputCol="FraudFound", outputCol="label")
>>> assembler = VectorAssembler(
...     inputCols=numeric_cols + [c + "_idx" for c in categorical_cols],
...     outputCol="features"
... )
>>> dt = DecisionTreeClassifier(labelCol="label", featuresCol="features", maxDepth=5)
>>> pipeline = Pipeline(stages=indexers + [label_indexer, assembler, dt])
>>> model = pipeline.fit(df)
>>> predictions = model.transform(df)
>>> predictions.select("FraudFound", "prediction").show(10)
+----------+----------+
|FraudFound|prediction|
+----------+----------+
|        No|       0.0|
|        No|       0.0|
|        No|       0.0|
|        No|       0.0|
|        No|       0.0|
|        No|       0.0|
|        No|       0.0|
|        No|       0.0|
|        No|       0.0|
|        No|       0.0|
+----------+----------+
only showing top 10 rows

>>> from pyspark.ml.evaluation import MulticlassClassificationEvaluator
>>> evaluator_precision = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="weightedPrecision")
>>> evaluator_recall = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="weightedRecall")
>>> precision = evaluator_precision.evaluate(predictions)
>>> recall = evaluator_recall.evaluate(predictions)
>>> print(f"Precision: {precision}")
Precision: 0.9458514979351691
>>> print(f"Recall: {recall}")
Recall: 0.9425421530479896
```

Evaluation & display decision tree

```
>>> from pyspark.ml.evaluation import MulticlassClassificationEvaluator
>>> evaluator_precision = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="weightedPrecision")
>>> evaluator_recall = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="weightedRecall")
>>> precision = evaluator_precision.evaluate(predictions)
>>> recall = evaluator_recall.evaluate(predictions)
>>> print(f"Precision: {precision}")
Precision: 0.9458514979351691
>>> print(f"Recall: {recall}")
Recall: 0.9425421530479896
>>> tree_model = model.stages[-1]
>>> print(tree_model.toDebugString)
DecisionTreeClassificationModel: uid=DecisionTreeClassifier_b368311754af, depth=5, numNodes=19, numClasses=2, numFeatures=32
  If (feature 31 in {1.0})
   If (feature 10 in {0.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0,12.0,13.0,14.0,15.0})
    Predict: 0.0
   Else (feature 10 not in {0.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0,12.0,13.0,14.0,15.0})
    If (feature 21 in {1.0})
     If (feature 0 <= 2.5)
      Predict: 1.0
     Else (feature 0 > 2.5)
      Predict: 0.0
    Else (feature 21 not in {1.0})
     If (feature 23 in {2.0,3.0,6.0,7.0})
      Predict: 0.0
     Else (feature 23 not in {2.0,3.0,6.0,7.0})
      If (feature 2 <= 23.5)
       Predict: 1.0
      Else (feature 2 > 23.5)
       Predict: 0.0
  Else (feature 31 not in {1.0})
   If (feature 16 in {1.0})
    If (feature 29 in {0.0,1.0,3.0})
     Predict: 0.0
    Else (feature 29 not in {0.0,1.0,3.0})
     If (feature 24 in {2.0,3.0,4.0,5.0})
      Predict: 0.0
     Else (feature 24 not in {2.0,3.0,4.0,5.0})
      Predict: 1.0
   Else (feature 16 not in {1.0})
    Predict: 0.0
```

Results & Conclusion

Algorithm Used: Decision Tree Classifier

Model Performance:

The performance of the Decision Tree model was evaluated using precision and recall metrics, which are crucial for assessing fraud detection accuracy.

Metric Value	value
Precision	**0.9458**
Recall	**0.9425**

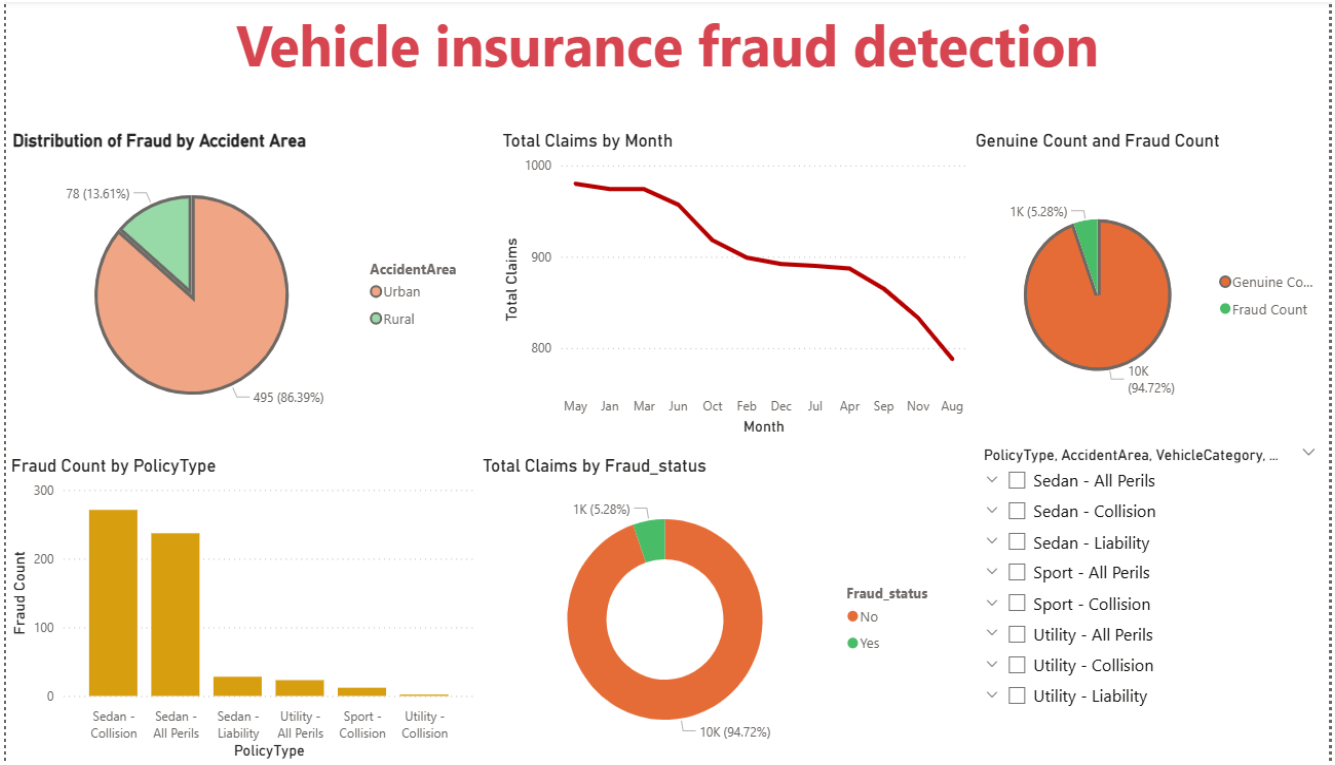
Conclusion:

The Decision Tree Classifier built using PySpark performed exceptionally well in detecting fraudulent vehicle insurance claims.
With a precision of ~94.6% and recall of ~94.2%, the model demonstrates strong predictive power and minimal false positives/negatives.

Key takeaways:

- PySpark efficiently handled categorical encoding, feature assembly, and model training on a large dataset.
- Decision Tree provided interpretable rules that can help insurance analysts understand fraud patterns.
- This project shows that machine learning can significantly enhance fraud detection processes, helping companies save time and money.

DASHBOARD



Power BI Dashboard showing fraud trends and distribution by region, policy, and time.

Dashboard Description and Insights

The Vehicle Insurance Fraud Detection Dashboard was developed in Microsoft Power BI to visually analyse fraud patterns and identify key factors contributing to fraudulent claims. The dashboard integrates multiple KPIs and visuals for a comprehensive understanding of claim behaviour.

Dashboard Components

1. **Distribution of Fraud by Accident Area (Pie Chart):**
   Displays the proportion of fraudulent claims between *Urban* and *Rural* areas.
   → *Insight:* Most fraudulent claims originate from Urban regions (≈86%), suggesting that fraud detection efforts should focus more on urban customers.

2. **Total Claims by Month (Line Chart):**
   Shows the monthly trend of total claims throughout the year.
   → *Insight:* The number of claims shows a declining trend over time, possibly due to improved fraud detection measures or seasonal variations in claim submissions.
3. **Genuine Count and Fraud Count (Pie Chart):**
   Compares the total number of genuine and fraudulent claims.
   → *Insight:* Fraudulent claims account for roughly 5% of total claims, indicating that while rare, they have a notable financial impact.
4. **Fraud Count by Policy Type (Bar Chart):**
   Illustrates which policy types are most affected by fraud.
   → *Insight:* Sedan–Collision and Sedan–All Perils policies report the highest number of fraud cases, making them high-risk categories.
5. **Total Claims by Fraud Status (Donut Chart):**
   Highlights the ratio of fraudulent ("Yes") to non-fraudulent ("No") claims.
   → *Insight:* The donut visualization clearly emphasizes the dominance of genuine claims while visually isolating fraudulent ones.
6. **Interactive Slicers:**
   Users can filter the dashboard by *Policy Type*, *Accident Area*, or *Vehicle Category* to explore fraud patterns dynamically

Final Insight:
The integration of PySpark for model training and Power BI for visualization created a complete end-to-end fraud detection system. This combination allows both predictive accuracy and visual interpretability, supporting data-driven decision-making for insurance firms.