Project report — Old-car accident classification

Executive summary

I converted a used-cars dataset into a machine-learning-ready table, engineered features (including missingness flags and groupings for rare categories), encoded categorical variables (one-hot and frequency), split data into train/val/test (stratified), trained an XGBoost classifier using SageMaker built-in XGBoost, and evaluated it locally using the trained model artifact. The model achieves AUC ≈ 0.7014. With a threshold chosen for high recall (≈0.3471), it reaches recall ≈ 0.80, precision ≈ 0.35, F1 ≈ 0.488. I iterated on preprocessing and thresholding rather than deploying an endpoint.

1. Environment & tooling

- Local Jupyter for EDA, preprocessing and feature engineering.
- SageMaker (console + notebook instance) for training with built-in XGBoost algorithm; S3 for data + model artifacts.

2. Data cleaning & parsing

I inspected columns and converted text fields into proper numeric/categorical formats so models can use them.

Price, mileage, model_year, engine

- **Price:** removed currency symbols and commas, then converted to numeric. This makes price comparable across rows.
- **Mileage:** stripped non-digits (commas, text like "km") then converted to numeric.
- **Model year:** forced to numeric.
- **Engine size:** parsed the engine text (e.g., 2.0L) and extracted the numeric part as engine_l (e.g., 2.0).
- I used **errors='coerce'** on conversions so bad values become NaN and are handled explicitly rather than causing crashes.

3. Target creation and missing-target handling

- I inspected accident values and created accident_label with 0 for “None reported” and 1 for “At least 1 accident or damage reported.”

- I dropped rows with missing accident labels because a supervised classifier needs labeled examples.

4. Missingness as a feature

- I created binary flags that record whether a column value was missing: e.g., clean_title_missing, fuel_type_missing, engine_l_missing (1 = missing, 0 = present).

- I checked whether missingness correlated with the accident label (cross-tabulations). Missingness often had different accident rates than non-missing rows.

5. Imputation (how I filled missing values)

- Categorical columns (clean_title, fuel_type, etc.): I filled missing entries with the string "Unknown". I kept side-by-side _missing flags so the model knows which "Unknown"es were originally missing.

- Numeric columns (engine_l): I filled missing values with the median of the column.

6. Categorical handling — rare groups & encoding

I divided categorical encoding into two strategies to keep the feature set manageable and meaningful.

6.1 Rare-grouping

- For very high-cardinality columns (especially model and ext_col), I combined categories that appear fewer than 10 times into a single bucket "Other".
- On this dataset:
    - model had many unique values → 1830 rare models were grouped into model_grp = "Other".
    - ext_col (exterior color) had many values → 298 rare colors were grouped into ext_col_grp = "Other".
    - Columns with fewer uniques (e.g., brand with 57 uniques) were left as-is.

6.2 Encoding after grouping

- **One-hot encoding (create binary columns for each category) for low-cardinality columns:** I applied this to fuel_type and clean_title. These have a small number of distinct values and one-hot is straightforward.
- **Frequency encoding for higher-cardinality columns:** I converted brand, model_grp, transmission, ext_col_grp, int_col into numeric features by replacing each category with its frequency (the fraction of rows having that value). Frequency encoding keeps representation compact and often conveys useful information about common vs rare categories.

7. Train/val/test split and class imbalance

- I split the dataset into train (70%), validation (15%) and test (15%) using stratified split to keep accident proportions consistent between train/test.
- I computed scale_pos_weight = (#negatives / #positives) from the training set (≈ 2.95) to inform XGBoost that accidents are the minority class.

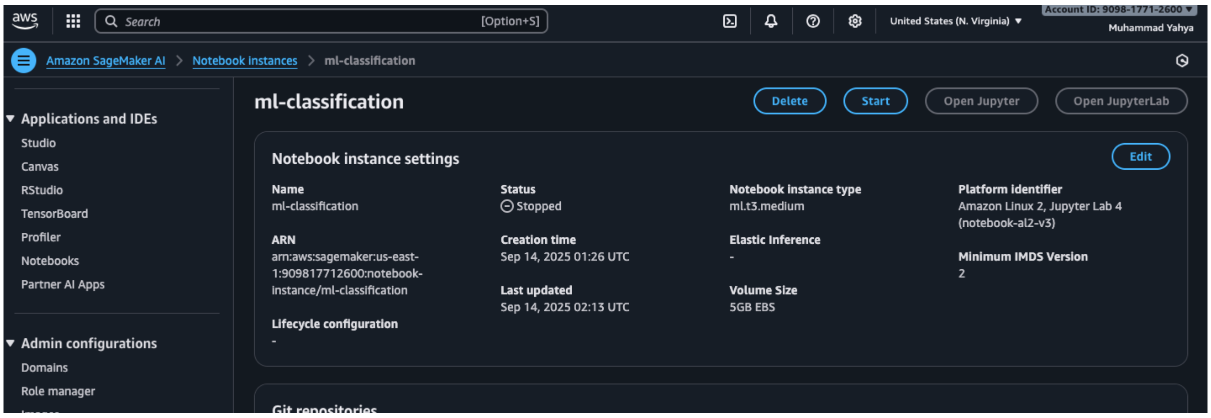
8. SageMaker Training

- I uploaded the preprocessed training and validation CSV files to S3 under my bucket/prefix.

- I launched a small SageMaker Notebook instance for the console and ran one training job using the built-in XGBoost container. I used a modest instance type (e.g., ml.t3.medium) and made sure to stop the notebook after launching training.

- I used XGBoost with objective=binary:logistic and eval_metric=auc. I ran a baseline job, examined logs, then submitted a tuned job with adjusted hyperparameters (lower learning rate, smaller max depth, modest regularization, and scale_pos_weight from the training set).

- The training job completed; the model artifact was saved to S3.



ml-classification

Notebook instance settings

Name
ml-classification

Status
Stopped

Notebook instance type
ml.t3.medium

Platform identifier
Amazon Linux 2, Jupyter Lab 4 (notebook-al2-v3)

ARN
arn:aws:sagemaker:us-east-1:909817712600:notebook-instance/ml-classification

Creation time
Sep 14, 2025 01:26 UTC

Elastic Inference
-

Minimum IMDS Version
2

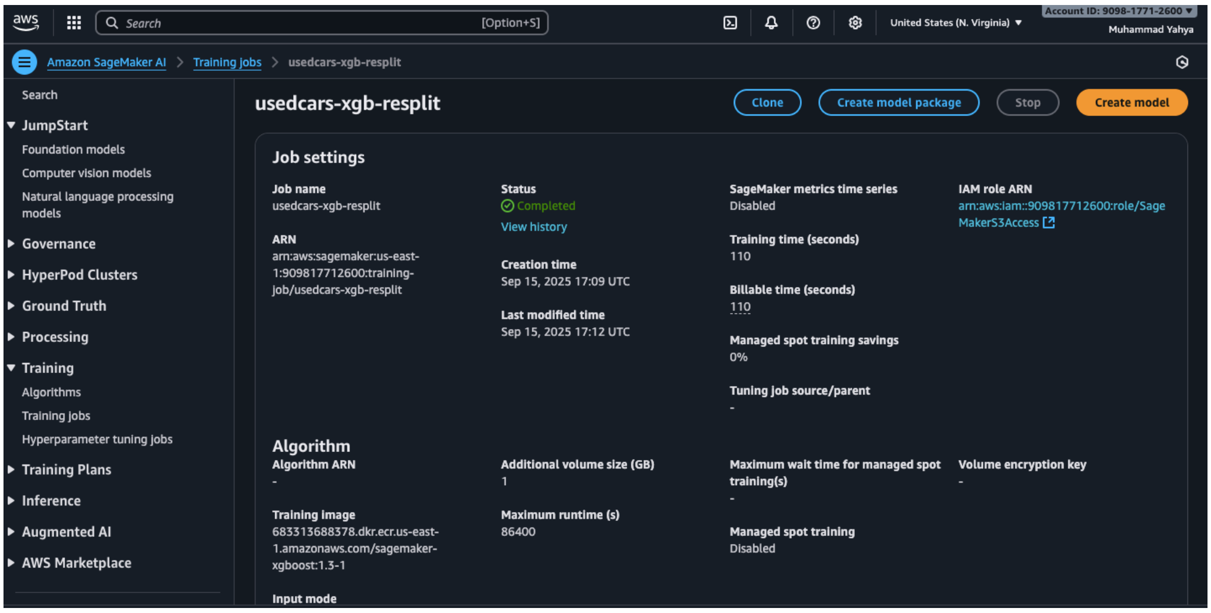
Last updated
Sep 14, 2025 02:13 UTC

Volume Size
5GB EBS

Lifecycle configuration
-

Git repositories

Figure 1 SageMaker Notebook Instance



usedcars-xgb-resplit

Job settings

Job name
usedcars-xgb-resplit

Status
Completed
View history

SageMaker metrics time series
Disabled

IAM role ARN
arn:aws:iam::909817712600:role/SageMakerS3Access

ARN
arn:aws:sagemaker:us-east-1:909817712600:training-job/usedcars-xgb-resplit

Training time (seconds)
110

Creation time
Sep 15, 2025 17:09 UTC

Billable time (seconds)
110

Last modified time
Sep 15, 2025 17:12 UTC

Managed spot training savings
0%

Tuning job source/parent
-

Algorithm

Algorithm ARN
-

Additional volume size (GB)
1

Maximum wait time for managed spot training(s)
-

Volume encryption key
-

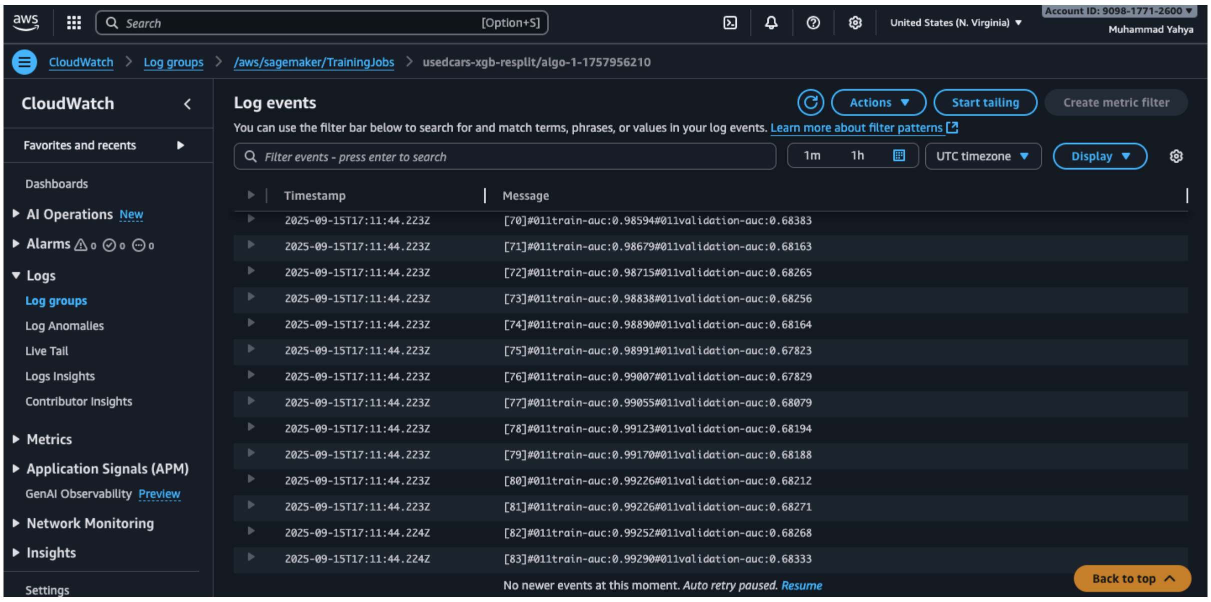
Training image
683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost:1.3-1

Maximum runtime (s)
86400

Managed spot training
Disabled

Input mode

Figure 2 SageMaker Training Job



Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                  Message
2025-09-15T17:11:44.223Z   [70]#011train-auc:0.98594#011validation-auc:0.68383
2025-09-15T17:11:44.223Z   [71]#011train-auc:0.98679#011validation-auc:0.68163
2025-09-15T17:11:44.223Z   [72]#011train-auc:0.98715#011validation-auc:0.68265
2025-09-15T17:11:44.223Z   [73]#011train-auc:0.98838#011validation-auc:0.68256
2025-09-15T17:11:44.223Z   [74]#011train-auc:0.98890#011validation-auc:0.68164
2025-09-15T17:11:44.223Z   [75]#011train-auc:0.98991#011validation-auc:0.67823
2025-09-15T17:11:44.223Z   [76]#011train-auc:0.99007#011validation-auc:0.67829
2025-09-15T17:11:44.223Z   [77]#011train-auc:0.99055#011validation-auc:0.68079
2025-09-15T17:11:44.223Z   [78]#011train-auc:0.99123#011validation-auc:0.68194
2025-09-15T17:11:44.223Z   [79]#011train-auc:0.99170#011validation-auc:0.68188
2025-09-15T17:11:44.223Z   [80]#011train-auc:0.99226#011validation-auc:0.68212
2025-09-15T17:11:44.223Z   [81]#011train-auc:0.99226#011validation-auc:0.68271
2025-09-15T17:11:44.224Z   [82]#011train-auc:0.99252#011validation-auc:0.68268
2025-09-15T17:11:44.224Z   [83]#011train-auc:0.99290#011validation-auc:0.68333
                           No newer events at this moment. Auto retry paused. Resume
```

Figure 3 CloudWatch Training Job Logs

9. Model artifact and local evaluation

- I downloaded model.tar.gz from S3, extracted it locally, and loaded the XGBoost model for evaluation.
- I loaded the preprocessed test CSV (that contains the true accident_label) and prepared the feature matrix matching the final_features ordering.
- I calculated predicted probabilities, then evaluated performance.
- Final evaluation results (key :
  - Test AUC: 0.7014 — the model ranks positives better than random.
  - With threshold chosen to satisfy recall ≥ 0.80 (threshold ≈ 0.3471):
    - Precision: 0.35
    - Recall: 0.8041
    - F1: 0.4877

- Confusion matrix:
  - True negatives: 216
  - False positives: 221
  - False negatives: 29
  - True positives: 119

10. Conclusion

In conclusion, I successfully built and evaluated a machine learning model to predict the likelihood of a used car having an accident history. The data required significant cleaning, feature engineering (including handling missing values, creating missingness flags, and encoding categorical variables), and careful preprocessing before training. I then trained and tested the model using SageMaker and later evaluated it locally. The final model achieved an AUC of 0.70, with good recall (0.80) for accident cases, meaning it was effective at identifying cars with accident history, though it came at the cost of lower precision (0.35). Overall, the project demonstrates a strong foundation for accident risk prediction and provides a scalable framework that can be further improved with more balanced data and feature refinements.