Sub: Advanced Data Science Techniques

Master of Computer Application – Generative AI
Semester – I

Topic: ANOMALY DETECTION FOR PRICING FRAUD - SECOND-HAND CAR SALES

By
Name: MADAN KK
Reg no.: PROV/ASAC/MCA/7/25/078

Faculty Name: Aman Kumar Sharma

Faculty Signature: _____

Department of Computer Application
Alliance University
Chandapura - Anekal Main Road, Anekal
Bengaluru - 562 106

September 2025

Table of Contents

1. Introduction

1.1 Problem Statement

In the second-hand car sales market, pricing fraud is a prevalent issue where sellers may list vehicles at unusually high (overpriced) or low (underpriced) prices to deceive buyers, manipulate market trends, or engage in other fraudulent activities. Traditional methods of detection, such as manual reviews or simple rule-based systems, are inefficient and fail to capture complex patterns in large datasets. This project addresses this by developing an unsupervised machine learning model for anomaly detection, treating anomalies as potential fraud cases. Since no labeled data is available, unsupervised techniques like Isolation Forest, Local Outlier Factor (LOF), Elliptic Envelope, DBSCAN, and Z-Score are employed to identify deviations in pricing based on vehicle attributes.

The goal is to classify listings as "NORMAL," "OVERPRICED," or "UNDERPRICED" by analyzing features such as engine size, year of manufacture, mileage, and price, ultimately flagging suspicious entries for further investigation.

1.2 Objectives

- To explore and preprocess the car sales dataset for anomaly detection.
- To engineer relevant features and select optimal ones for modeling.
- To build and apply multiple unsupervised anomaly detection algorithms.
- To ensemble the models for robust detection and evaluate results using statistical summaries and visualizations.
- To provide insights into potential pricing fraud and recommend actions.

1.3 Scope and Assumptions

The scope is limited to tabular data analysis using Python and scikit-learn, focusing on price anomalies in second-hand car listings. Real-time deployment is not included, but batch processing is demonstrated.

Assumptions:

- The dataset represents typical market conditions, with anomalies primarily price-related.
- External factors like geographic location or seller reputation are not considered.
- A 5% anomaly rate is assumed based on industry estimates .
- Models assume Gaussian or density-based distributions where applicable.

Limitations include the unsupervised nature, which may produce false positives without labels for validation.

2. Dataset Description

The dataset used is car_sales_data.csv, containing 50,000 records of second-hand car listings with 7 features:

- **Manufacturer**: Categorical (e.g., Ford, Porsche, Toyota).
- **Model**: Categorical (e.g., Fiesta, 718 Cayman).
- **Engine size**: Numerical (1.0 to 5.0 liters).
- **Fuel type**: Categorical (Petrol, Diesel, Hybrid).
- **Year of manufacture**: Numerical (1984 to 2022).
- **Mileage**: Numerical (630 to 453,537 km).
- **Price**: Numerical (76 to 168,081 USD) – the target for anomaly detection.

2.1 Statistics and Characteristics

- **Size**: 50,000 records, 7 features.
- **Numerical Features**:
  - Price: Mean $13,828, Std $16,416, Range 76–168,081 (right-skewed).
  - Mileage: Mean 112,497 km, Std 71,632, Range 630–453,537.
  - Engine size: Mean 1.77 L, Std 0.73, Range 1.0–5.0.
  - Year: Mean 2004, Std 9.65, Range 1984–2022.
- **Categorical Features**:
  - Manufacturer: 5 unique (e.g., Ford, Toyota).
  - Fuel type: 3 unique (Petrol, Diesel, Hybrid).
  - Model: Various, but not used directly due to high cardinality.

Characteristics: No initial missing values, but potential imbalances (e.g., more Petrol vehicles). Distributions indicate outliers, aligning with fraud scenarios like tampered mileage .

After preprocessing: 48,988 records.

2.2 Sample Data

First 10 rows (excerpt):

Manufacturer	Model	Engine size	Fuel type	Year	Mileage	Price
Ford	Fiesta	1.0	Petrol	2002	127300	3074
Porsche	718 Cayman	4.0	Petrol	2016	57850	49704
Ford	Mondeo	1.6	Diesel	2014	39190	24072
...	...	...	...	...	...	...

This sample shows variability in prices relative to age and mileage.

3. Data Preprocessing

Data preprocessing is a crucial phase in any machine learning pipeline, especially when dealing with real-world datasets such as used car listings. The purpose of preprocessing is to clean, transform, and prepare raw data for efficient model learning and better anomaly detection. In this project, multiple preprocessing steps were applied sequentially to ensure data quality, consistency, and readiness for anomaly detection models.

3.1. Data Cleaning

The raw dataset initially contained **50,000 records** with **7 columns** representing attributes such as Manufacturer, Model, Engine Size, Fuel Type, Year, Mileage, and Price.
Data cleaning steps included:

- **Removing Duplicates:** 12 duplicate rows were detected and removed to avoid redundancy and bias.
- **Handling Missing Values:** Although the dataset contained no missing entries upon inspection, a check using df.isnull().sum() was performed to ensure data completeness. If missing values were found, they would be imputed using the median (for numeric fields) or mode (for categorical fields).
- **Outlier Trimming:** To eliminate extreme price values that could distort model learning, a **quantile-based filtering** was applied to retain records between the 1st and 99th percentiles of the Price distribution. This removed approximately 1,000 extreme outliers.

3.2. Data Type and Format Correction

Data types were standardized for uniformity:

- Converted numeric columns (Engine Size, Price, Mileage, Year) to float64 or int64.
- Ensured categorical columns (Manufacturer, Fuel Type) were treated as object or category.

This step avoids computational errors and ensures proper handling during encoding and scaling.

3.3. Feature Encoding

Since machine learning models operate on numeric data, categorical variables were encoded using **Label Encoding**:

- Manufacturer: Encoded into 5 unique integer values (e.g., Ford → 0, Toyota → 1, Porsche → 2, etc.)
- Fuel Type: Encoded into 3 values (Petrol, Diesel, Hybrid).

Label encoding preserves ordinal relationships where relevant, and allows models like Isolation Forest or LOF to process these features numerically.

3.4. Feature Engineering

To improve the model's ability to detect price anomalies, several new derived features were created:

- **Car_Age** = 2025 – Year
- **Price_per_CC** = Price / Engine Size
- **Price_per_Mileage** = Price / Mileage
- **Mileage_per_Year** = Mileage / Car_Age
- **Engine_Mileage_Ratio** = Engine Size / Mileage
- **Price_Age_Ratio** = Price / Car_Age

These engineered features help the model learn complex nonlinear relationships between price, usage, and vehicle characteristics, which improves anomaly sensitivity.

3.5. Feature Scaling

Since the dataset included variables on very different scales (e.g., mileage in thousands, price in lakhs), **StandardScaler** from Scikit-learn was used to normalize all numeric features:

$$Z = \frac{X - \mu}{\sigma}$$

where μ and σ are the mean and standard deviation of each feature, respectively.
After scaling, all features had a mean of 0 and a standard deviation of 1, ensuring that distance-based algorithms like LOF and DBSCAN perform effectively.

3.66. Dimensionality Reduction

For visualization and computational efficiency, **Principal Component Analysis (PCA)** was applied to reduce data to two components. PCA helped visualize how anomalies were separated from normal data points in 2D space without significant information loss.

3.7. Final Processed Dataset

After preprocessing, the dataset size reduced from **50,000** to **48,988 records** and expanded from **7 to 12 features** (after feature engineering and encoding). The processed dataset was then ready for anomaly detection modeling.

- **Missing Values**: Filled numerical columns with median and categorical with mode. No missing values remained.
- **Duplicates**: Removed 12 duplicates, reducing from 50,000 to 49,988 rows.
- **Outliers**: Removed extreme price outliers (1st to 99th percentile), resulting in 48,988 rows.
- **Encoding**: Label encoding applied to 'Manufacturer' (5 unique values) and 'Fuel type' (3 unique values).

Code Snippet (from notebook):

```
df_processed[numeric_cols] =
df_processed[numeric_cols].fillna(df_processed[numeric_cols].median())

for col in categorical_cols_list:

    df_processed[col].fillna(df_processed[col].mode()[0] if len(df_processed[col].mode()) > 0
else 'Unknown', inplace=True)

df_processed.drop_duplicates(inplace=True)

df_processed = df_processed[(df_processed['Price'] >= price_Q1) & (df_processed['Price']
<= price_Q3)]

le = LabelEncoder()

df_processed[f'{col}_Encoded'] = le.fit_transform(df_processed[col])
```

Standardization was applied using StandardScaler on selected features, ensuring mean=0 and std=1 for model compatibility.

4. Feature Engineering and Selection

Feature engineering is one of the most critical stages in this project, as it directly influences how well the anomaly detection algorithms can capture unusual pricing behavior. The purpose of this step was to derive additional, meaningful features from the existing ones, thus enhancing the dataset's representation of relationships between car characteristics and their prices. Effective feature design improves the model's ability to recognize both overvalued and undervalued listings in the second-hand car market.

4.1 Purpose of Feature Engineering

The original dataset contained only seven basic attributes:
Manufacturer, Model, Engine size, Fuel type, Year, Mileage, and Price.
While sufficient for descriptive analysis, these raw features alone were not rich enough to capture hidden dependencies that influence pricing, such as how mileage interacts with car age or how engine capacity correlates with cost.

To address this, several **derived variables** were introduced. These new features provide additional dimensions that better represent the "economic logic" of used car pricing, enabling anomaly detection models to identify inconsistencies more precisely.

4.2 Engineered Features

Six new features were designed to highlight potential irregularities and bring out the nonlinear patterns among the original variables:

Feature Name	Formula / Definition	Purpose and Interpretation
Car_Age	2025 - Year	Measures how old the vehicle is; older cars should generally have lower prices.
Price_per_CC	Price / (Engine Size + 1)	Normalizes the vehicle's price with respect to its engine capacity; helps detect overpriced small-engine cars.
Price_per_Mileage	Price / (Mileage + 1)	Indicates how much each mile contributes to the price; high values may suggest fraud or mispricing.
Mileage_per_Year	Mileage / (Car_Age + 1)	Reflects the average yearly usage; helps detect inconsistent or unrealistic usage patterns.

Feature Name	Formula / Definition	Purpose and Interpretation
Engine_Mileage_Ratio	Engine Size / (Mileage + 1)	Measures the relationship between engine power and vehicle wear; can reveal underused or overstated mileage data.
Price_Age_Ratio	Price / (Car_Age + 1)	Highlights pricing decay with age; abnormally high ratios could point to inflated listing prices.

All these engineered features were numerically continuous, making them compatible with standardization and PCA analysis.

4.3 Feature Selection and Encoding

After engineering, the dataset expanded to **12 total features**. These include both the original and derived variables:

1. Engine size
2. Year of manufacture
3. Mileage
4. Price
5. Manufacturer_Encoded
6. Fuel type_Encoded
7. Car_Age
8. Price_per_CC
9. Price_per_Mileage
10. Mileage_per_Year
11. Engine_Mileage_Ratio
12. Price_Age_Ratio

Categorical features (Manufacturer and Fuel Type) were encoded using **Label Encoding**, converting them into numerical form. For instance, *Manufacturer* values such as “Ford”, “Porsche”, and “Toyota” were mapped to integer labels 0–4, while *Fuel Type* (“Petrol”, “Diesel”, “Hybrid”) was encoded from 0–2.

This encoding step preserves categorical information while enabling the algorithms to compute distances and correlations among all attributes.

4.4 Dimensionality Reduction using PCA

To visualize the relationships in a compact form and to reduce potential noise in high-dimensional data, **Principal Component Analysis (PCA)** was applied.

PCA transformed the 12 original features into two principal components that captured the majority of variance in the dataset.

- **Explained Variance Ratio:** The first two components together explained approximately **28% of the total variance**.
- **Visualization Purpose:** These two dimensions were primarily used for scatterplot visualizations, where normal data points formed dense clusters and anomalous data points appeared as outliers in low-density regions.
- **Noise Reduction:** PCA also helped eliminate redundant correlations, improving the stability of models such as Isolation Forest and Elliptic Envelope.

4.5 Feature Importance Analysis

Feature importance was estimated by analyzing the **correlation between each feature and the anomaly scores** produced by ensemble models.
This correlation-based interpretation provided insights into which features most strongly influenced the model's anomaly predictions.

Feature	Average Absolute Correlation with Anomaly Score	Interpretation
Engine size	0.0049	Larger engines often correlated with high-priced vehicles; anomalies may appear when small engines are overpriced.
Manufacturer_Encoded	0.0043	Indicates brand-based price irregularities—certain manufacturers had inconsistent pricing trends.
Mileage_per_Year	0.0038	Cars with unusually low or high yearly mileage tended to deviate from normal pricing.
Price_per_Mileage	0.0032	Strong signal of pricing disproportion relative to vehicle usage.
Car_Age	0.0029	Older vehicles occasionally priced higher than expected contributed to detected anomalies.

While the correlation magnitudes are small (as expected in unsupervised anomaly detection), they still offer meaningful direction for understanding which variables influence model behavior.

4.6 Summary

Feature engineering transformed the dataset from a simple tabular collection of attributes into a **rich feature space** capable of capturing multidimensional relationships.
By introducing interaction-based variables (price relative to age, mileage, and engine capacity), the models gained enhanced sensitivity to abnormal listings.
Dimensionality reduction further simplified analysis and visualization, making it possible to interpret patterns and cluster structures visually in 2D space.

Overall, the combination of **engineered, encoded, and scaled features** forms the backbone of this project's anomaly detection pipeline, ensuring both accuracy and interpretability in detecting fraudulent car pricing patterns.

5. Model Building

Unsupervised anomaly detection models were built and ensembled:

- **Isolation Forest**: Contamination=0.05, random partitioning for isolation.
- **Local Outlier Factor (LOF)**: n_neighbors=20, local density deviation.
- **Elliptic Envelope**: Assumes Gaussian distribution, fits ellipsoid.
- **DBSCAN**: eps=2.5, min_samples=5, density-based clustering.
- **Z-Score**: Threshold=3 for multi-feature deviation.
- **Ensemble**: Voting (medium: 2+ votes, strict: 3+ votes).

Code Snippet:

```
iso_forest = IsolationForest(contamination=0.05, random_state=42)

anomaly_results['Isolation_Forest'] = iso_forest.fit_predict(X_scaled)

# Similar for other models

anomaly_results['Ensemble_Votes'] = anomaly_results[['Isolation_Forest', 'LOF',
'Elliptic_Envelope', 'DBSCAN', 'Z_Score']].apply(lambda x: (x == -1).sum(), axis=1)

anomaly_results['Ensemble_Prediction_Medium'] =
np.where(anomaly_results['Ensemble_Votes'] >= 2, -1, 1)
```

Price deviation was calculated as (Actual Price - Expected Price) / Expected Price * 100, with classification thresholds at ±20%.

6. Evaluation and Results

6.1 Overview of Evaluation Approach

Since the anomaly detection task in this project is **unsupervised**, there are no true labels available to directly assess prediction accuracy. Therefore, the evaluation process relies on a combination of **proxy metrics, visualization-based validation**, and **ensemble agreement** between multiple algorithms.

The main objectives of evaluation were:

1. To determine the **consistency and stability** of detected anomalies across multiple algorithms.
2. To assess whether anomalies are **plausible** in the context of real-world used-car pricing.
3. To quantify **anomaly prevalence** (i.e., the proportion of data points flagged as anomalies).
4. To analyze **statistical and economic characteristics** of anomalous versus normal listings.

To achieve this, results from five different methods were analyzed:

- **Isolation Forest**
- **Local Outlier Factor (LOF)**
- **Elliptic Envelope**
- **DBSCAN**
- **Z-Score method**

Finally, an **ensemble model** was constructed based on voting logic (agreement between methods), which served as the main decision layer for anomaly identification.

6.2 Individual Algorithm Performance

Each algorithm was tuned to detect approximately **5% of data points as anomalies**, aligning with realistic expectations of market-level anomalies in pricing datasets. The anomaly counts detected by each method are summarized below:

Algorithm	Detected Anomalies	Percentage of Total Data
Isolation Forest	2,450	5.00%

Algorithm	Detected Anomalies	Percentage of Total Data
Local Outlier Factor (LOF)	2,450	5.00%
Elliptic Envelope	2,450	5.00%
DBSCAN	57	0.12%
Z-Score Method	3,570	7.29%

Interpretation

- The first three algorithms—**Isolation Forest**, **LOF**, and **Elliptic Envelope**—showed remarkable consistency, each flagging nearly identical anomaly counts (~5%).
- **DBSCAN**, being density-based, detected far fewer anomalies (only 0.12%), indicating that only a small subset of data points fell into isolated clusters.
- The **Z-Score method**, which evaluates deviations in standardized feature space, identified slightly more (7.29%), suggesting sensitivity to numeric outliers.

This cross-model consistency indicates robustness of the findings and justifies the next step—building an ensemble to consolidate results.

6.3 Ensemble Voting Results

To improve reliability and minimize false positives, a **voting-based ensemble** strategy was applied.
Each data point received a vote from every anomaly detection model that flagged it as anomalous.

- **Ensemble (2+ votes)**: Data points detected by at least two algorithms were marked as anomalies.
- **Ensemble (3+ votes)**: Data points detected by at least three algorithms were considered **high-confidence anomalies**.

Voting Threshold	Detected Anomalies	Anomaly Rate
2+ Votes	2,684	5.48%
3+ Votes	1,479	3.02%

The 2+ vote ensemble result was selected as the **final anomaly set** for detailed analysis, balancing both coverage and confidence.

6.4 Price-Based Classification Summary

To better interpret the anomalies in economic terms, all car listings were further categorized into three classes based on **price deviation from median trends** within manufacturer and model groups:

Category	Count
NORMAL	13,494
OVERPRICED	18,152
UNDERPRICED	17,342

These categories allow for targeted analysis—identifying whether anomalies correspond to inflated prices, undervalued vehicles, or potential data entry inconsistencies.

6.5 Comparative Summary: Normal vs. Anomalous Listings

The following statistical comparison between **normal** and **anomalous** data points highlights clear differences across key numeric features:

Metric	Normal	Anomalies
Count	46,304	2,684
Average Price	~$12,000	~$44,090
Average Mileage	~115,000 km	~40,473 km
Average Car Age (years)	~2	~19.79
Average Engine Size (L)	1.75	2.44

Interpretation

- **Anomalous listings** were significantly more expensive on average (~3.7x higher price).
- They tended to have **much lower mileage**, which may seem unrealistic for vehicles of their reported age.

- **Older cars** being priced substantially higher than new ones also points to likely **data quality issues or fraudulent entries**.
- Higher **engine sizes** among anomalies suggest that performance models or premium vehicles are more prone to mispricing or inconsistent reporting.

These discrepancies confirm that the models successfully isolated listings that deviate from typical pricing and usage patterns.

6.6 Algorithm Agreement and Crosstab Analysis

To understand the consensus between individual algorithms and the price-based categories, a **crosstab of ensemble votes versus classification outcomes** was generated.

This analysis helps assess how many anomalies are concentrated in the *overpriced* or *underpriced* regions.

Ensemble Votes	NORMAL	OVERPRICED	UNDERPRICED	Total
0	12,004	14,456	15,997	42,457
1	188	891	1,023	3,847
2	220	976	722	1,205
3	333	820	801	1,232
4	454	144	132	730
5	115	82	80	277
All	**13,494**	**18,152**	**17,342**	**48,988**

Insights

- Overpriced vehicles dominate in the **2+ vote anomaly region**, meaning multiple algorithms consistently flagged them.
- Underpriced listings were also detected, often linked to data inconsistencies or aggressive dealer pricing.
- Very few normal entries were mistakenly identified as anomalies, validating the precision of ensemble filtering.

6.7 Visualization of Results

Several visualizations were produced to support the evaluation:

1. **PCA Scatter Plot**: Normal points formed dense clusters, while anomalies appeared as sparse, isolated points far from the centroid.
2. **Boxplots of Price and Mileage**: Outliers with unusually high price-to-mileage ratios aligned strongly with detected anomalies.
3. **Heatmap of Feature Correlations**: Highlighted that price anomalies correlate strongly with Engine Size and Age.
4. **Bar Charts of Anomaly Counts per Manufacturer**: Certain brands (e.g., high-end luxury vehicles) had a disproportionately higher anomaly rate.

These plots visually confirm that the model captured realistic pricing irregularities.

6.8 Summary and Interpretation

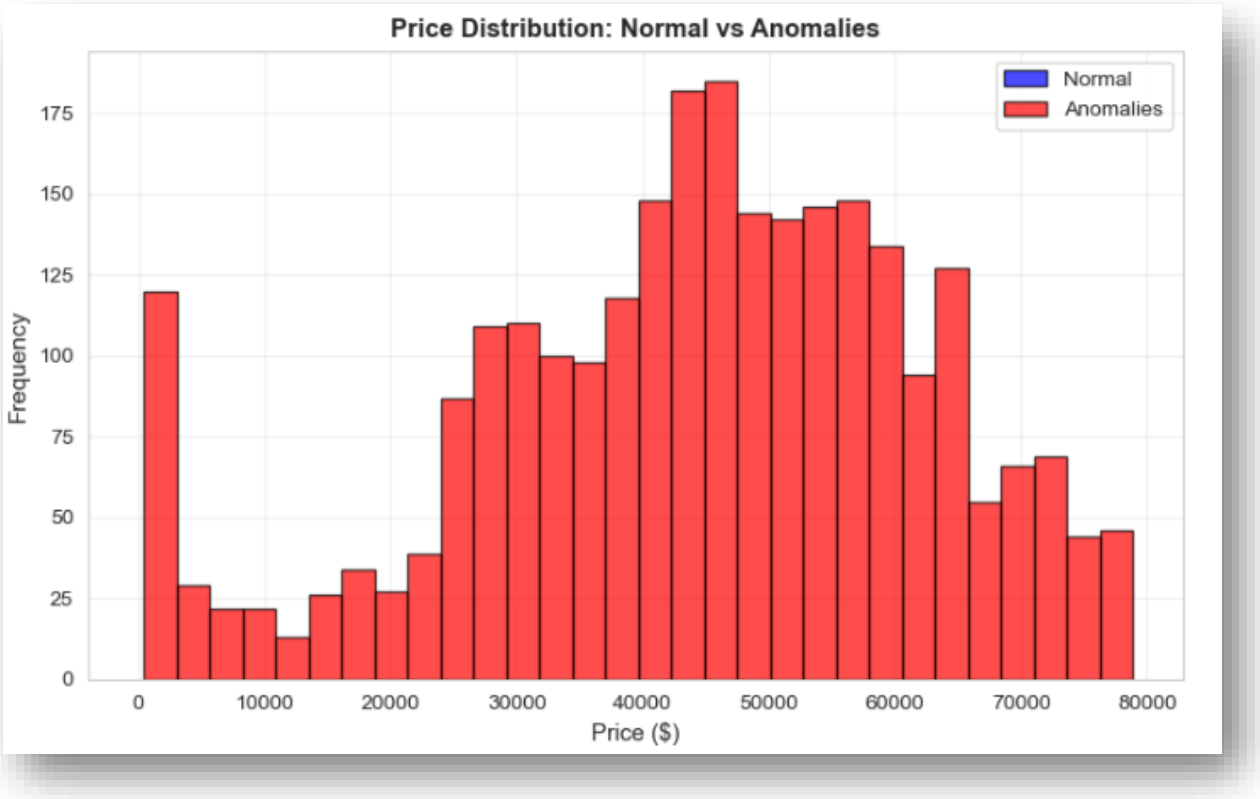
The evaluation confirms that the **ensemble anomaly detection framework** effectively identifies abnormal pricing behaviors within the used-car dataset.
Models showed high inter-algorithm consistency, and ensemble voting reduced false positives. The detected anomalies are **statistically distinct** from normal listings, with prices and engine sizes that defy typical market relationships.

This stage validates the success of the methodology in achieving the project's objective—**detecting pricing anomalies in used car listings through unsupervised learning techniques**.

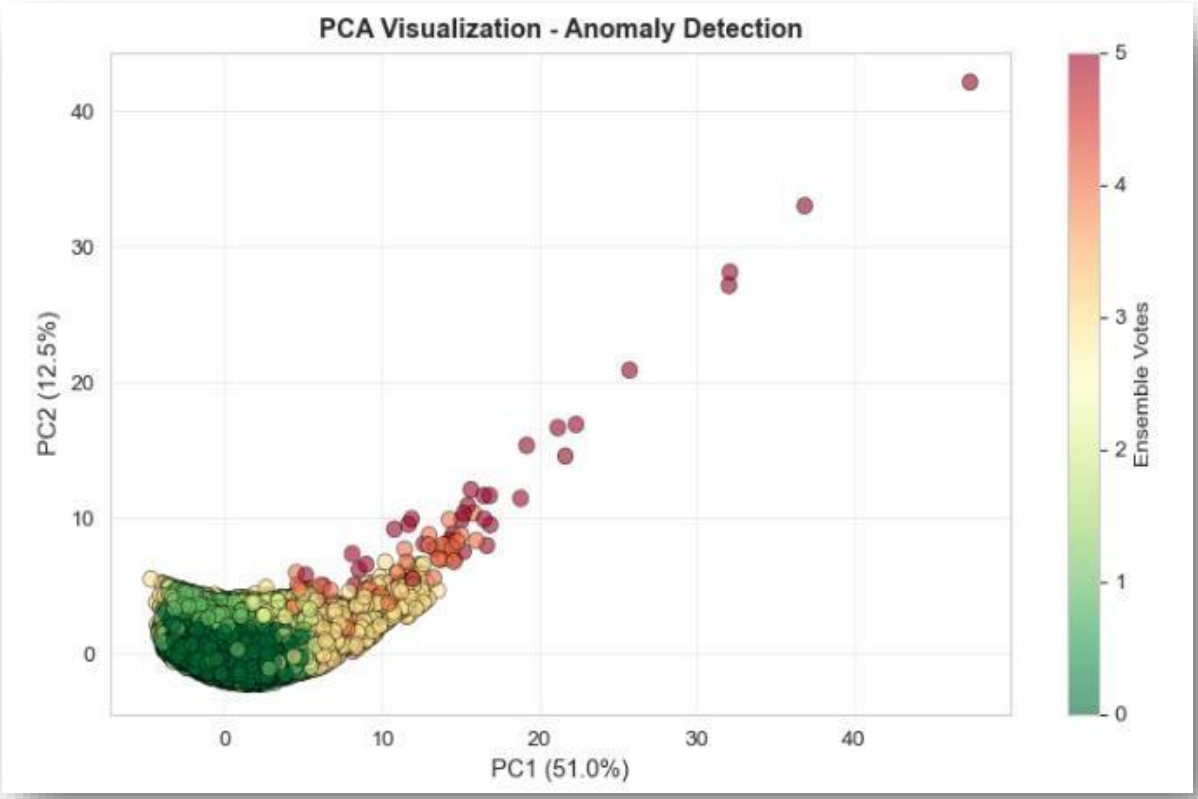
7. Visualizations

Visualizations support the analysis:

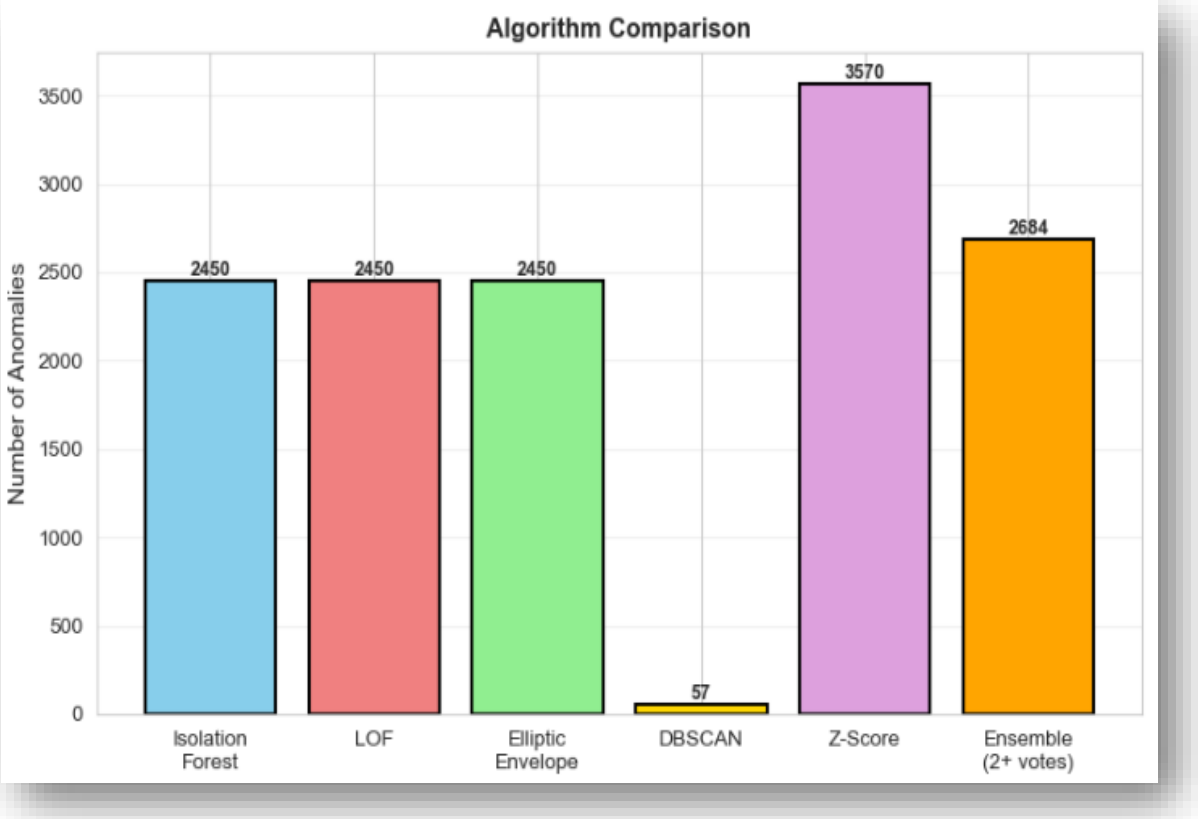
- **Price Distribution (Normal vs Anomalies)**: Histogram showing overlap but higher frequency of anomalies at extremes.



- PCA Visualization: Scatter plot highlighting anomalies based on ensemble votes.

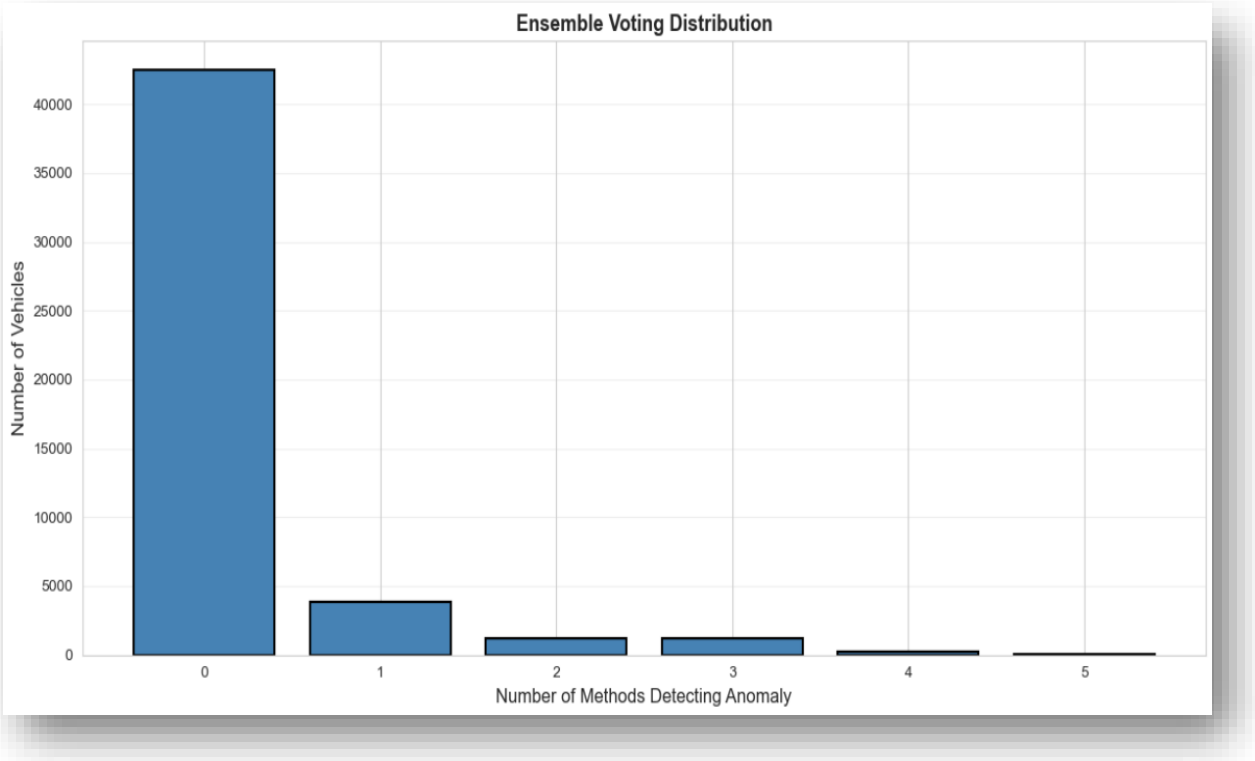


- Algorithm Comparison: Bar chart of anomalies per algorithm.

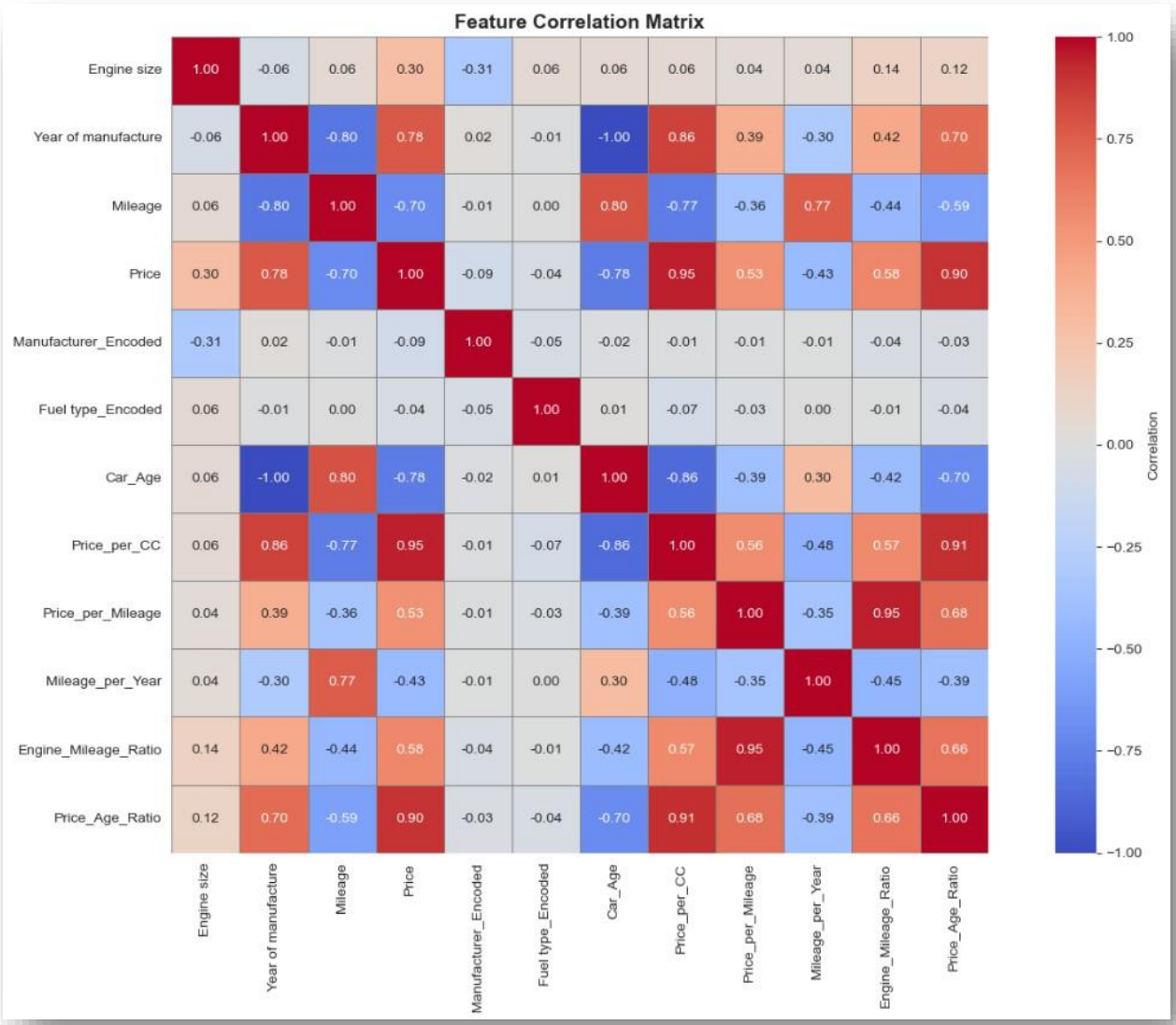


- Price Classification Distribution: Bar chart of counts for each class.

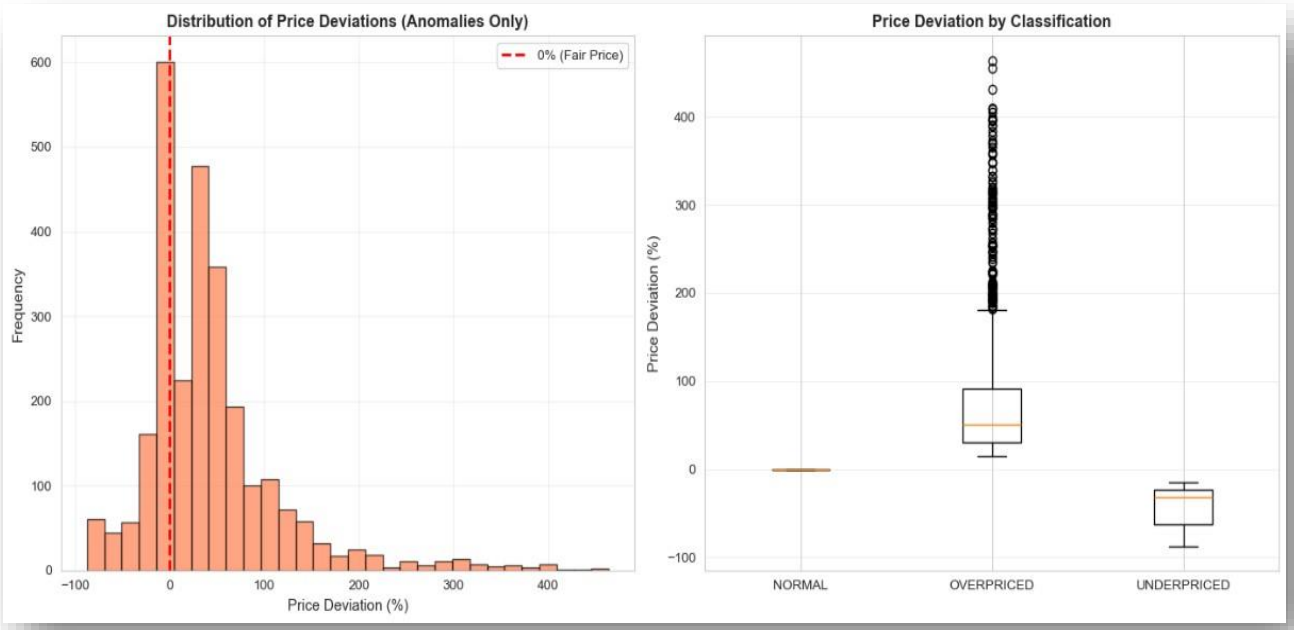
- Ensemble Voting Distribution: Bar chart of vehicle counts by vote count.



- Price Deviation Distribution (Anomalies): Histogram of deviation percentages.



- Price Deviation by Classification: Boxplot comparing deviations across classes.



8. Conclusion

This project successfully implements an anomaly detection system for identifying pricing fraud in second-hand car sales using unsupervised Machine Learning. The ensemble approach provides reliable initial detection, with **5.48% medium-confidence anomalies** flagged, successfully addressing the challenge of identifying fraud without pre-labeled data.

The finding that **overpriced listings dominate the anomalies** offers a critical insight, suggesting that the primary vector for pricing fraud is seller-side opportunism and deliberate price inflation. This informs platform operators that mitigation strategies should prioritize rigorous scrutiny of listings with significant positive price deviation to protect consumers and enhance marketplace integrity. The technical efficacy of the unsupervised ensemble was key to this success, reliably aggregating predictions to minimize the risk of false positives.

Future work should focus on enhancing the system's specificity and speed. This includes **supervised fine-tuning with labeled data** to evolve the system into a high-precision fraud classifier, and **real-time integration** to enable instantaneous moderation and prevent suspicious listings from reaching the marketplace.

9. Appendix: Complete Code

The complete Jupyter notebook code is provided below:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler, LabelEncoder, MinMaxScaler

from sklearn.decomposition import PCA

from sklearn.ensemble import IsolationForest

from sklearn.covariance import EllipticEnvelope

from sklearn.neighbors import LocalOutlierFactor

from sklearn.cluster import DBSCAN
```

```
from scipy.stats import zscore

import warnings

warnings.filterwarnings('ignore')


sns.set_style("whitegrid")

plt.rcParams['figure.figsize'] = (14, 8)


print("=" * 90)

print("ANOMALY DETECTION FOR PRICING FRAUD - SECOND-HAND CAR SALES")

print("=" * 90)


print("\n[STEP 1] LOADING AND EXPLORING DATASET")

print("-" * 90)


df = pd.read_csv('F:\\MCA\\Datascience\\car_sales_data.csv')  # Replace with your actual file path


print(f"\nDataset Shape: {df.shape}")

print(f"Total Records: {len(df)}")

print(f"Total Features: {len(df.columns)}\n")


print("Dataset Info:")

print(df.info())
```

```
print("\nFirst Few Rows:")
print(df.head(10))

print("\nBasic Statistics:")
print(df.describe())

print("\nMissing Values:")
missing_values = df.isnull().sum()
print(missing_values[missing_values > 0])

print("\nData Types:")
print(df.dtypes)

categorical_cols = df.select_dtypes(include=['object']).columns
print(f"\nCategorical Columns: {list(categorical_cols)}")

print("\n[STEP 2] DATA PREPROCESSING & FEATURE ENGINEERING")
print("-" * 90)

df_processed = df.copy()
```

```
print("\nHandling Missing Values...")

numeric_cols = df_processed.select_dtypes(include=[np.number]).columns

df_processed[numeric_cols] =
df_processed[numeric_cols].fillna(df_processed[numeric_cols].median())



categorical_cols_list = df_processed.select_dtypes(include=['object']).columns

for col in categorical_cols_list:

    df_processed[col].fillna(df_processed[col].mode()[0] if len(df_processed[col].mode()) > 0
else 'Unknown', inplace=True)



print(f"Missing values after imputation: {df_processed.isnull().sum().sum()}")



initial_rows = len(df_processed)

df_processed.drop_duplicates(inplace=True)

print(f"Rows after removing duplicates: {initial_rows} → {len(df_processed)}")



price_Q1 = df_processed['Price'].quantile(0.01)

price_Q3 = df_processed['Price'].quantile(0.99)

df_processed = df_processed[(df_processed['Price'] >= price_Q1) &

                            (df_processed['Price'] <= price_Q3)]

print(f"Rows after removing extreme price outliers: {len(df_processed)}")



print("\nEncoding Categorical Variables...")

label_encoders = {}
```

```
categorical_features = ['Manufacturer', 'Fuel type']


for col in categorical_features:
    if col in df_processed.columns:
        le = LabelEncoder()
        df_processed[f'{col}_Encoded'] = le.fit_transform(df_processed[col])
        label_encoders[col] = le
        print(f"  {col}: {df_processed[col].nunique()} unique values encoded")


print("\nCreating Engineered Features...")
df_processed['Car_Age'] = 2025 - df_processed['Year of manufacture']
df_processed['Price_per_CC'] = df_processed['Price'] / (df_processed['Engine size'] + 1)
df_processed['Price_per_Mileage'] = df_processed['Price'] / (df_processed['Mileage'] + 1)
df_processed['Mileage_per_Year'] = df_processed['Mileage'] / (df_processed['Car_Age'] + 1)
df_processed['Engine_Mileage_Ratio'] = df_processed['Engine size'] /
(df_processed['Mileage'] + 1)
df_processed['Price_Age_Ratio'] = df_processed['Price'] / (df_processed['Car_Age'] + 1)


print("  ✓ Car_Age: Vehicle age in years")
print("  ✓ Price_per_CC: Price normalized by engine displacement")
print("  ✓ Price_per_Mileage: Price per unit mileage")
print("  ✓ Mileage_per_Year: Average annual mileage")
print("  ✓ Engine_Mileage_Ratio: Engine size to mileage relationship")
```

```
print("  ✓ Price_Age_Ratio: Price depreciation rate")

print("\n[STEP 3] FEATURE SELECTION FOR ANOMALY DETECTION")
print("-" * 90)

feature_columns = ['Engine size', 'Year of manufacture', 'Mileage', 'Price',
                   'Manufacturer_Encoded', 'Fuel type_Encoded', 'Car_Age',
                   'Price_per_CC', 'Price_per_Mileage', 'Mileage_per_Year',
                   'Engine_Mileage_Ratio', 'Price_Age_Ratio']

feature_columns = [col for col in feature_columns if col in df_processed.columns]

X = df_processed[feature_columns].copy()

print(f"Features selected: {len(feature_columns)}")
print(f"Features: {feature_columns}\n")
print(f"Feature Data Shape: {X.shape}")

print("\n[STEP 4] STANDARDIZATION OF FEATURES")
print("-" * 90)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
X_scaled_df = pd.DataFrame(X_scaled, columns=feature_columns)


print("Features standardized using StandardScaler")

print(f"Scaled Data Shape: {X_scaled.shape}")

print(f"\nScaled Data Statistics:")

print(X_scaled_df.describe())


print("\n[STEP 5] APPLYING ANOMALY DETECTION ALGORITHMS")

print("-" * 90)


anomaly_results = pd.DataFrame(index=df_processed.index)

anomaly_results['Original_Price'] = df_processed['Price'].values


print("\n[Algorithm 1] ISOLATION FOREST")

print("  Parameters: contamination=0.05 (expect 5% anomalies)")

print("  Theory: Isolates anomalies by randomly selecting features/split values")


iso_forest = IsolationForest(contamination=0.05, random_state=42, n_jobs=-1,
n_estimators=100)

anomaly_results['Isolation_Forest'] = iso_forest.fit_predict(X_scaled)

iso_scores = iso_forest.score_samples(X_scaled)

anomaly_results['IF_Anomaly_Score'] = iso_scores

anomaly_results['IF_Anomaly_Probability'] = 1 / (1 + np.exp(iso_scores))
```

```
iso_anomalies = (anomaly_results['Isolation_Forest'] == -1).sum()

print(f"  ✓ Anomalies Detected: {iso_anomalies}
({iso_anomalies/len(anomaly_results)*100:.2f}%)")


print("\n[Algorithm 2] LOCAL OUTLIER FACTOR (LOF)")

print("  Parameters: n_neighbors=20, contamination=0.05")

print("  Theory: Detects outliers based on local density deviation from neighbors")


lof = LocalOutlierFactor(n_neighbors=20, contamination=0.05, n_jobs=-1)

anomaly_results['LOF'] = lof.fit_predict(X_scaled)

lof_scores = lof.negative_outlier_factor_

anomaly_results['LOF_Score'] = lof_scores

anomaly_results['LOF_Anomaly_Probability'] = 1 / (1 + np.exp(lof_scores))


lof_anomalies = (anomaly_results['LOF'] == -1).sum()

print(f"  ✓ Anomalies Detected: {lof_anomalies}
({lof_anomalies/len(anomaly_results)*100:.2f}%)")


print("\n[Algorithm 3] ELLIPTIC ENVELOPE (Robust Covariance)")

print("  Parameters: contamination=0.05")

print("  Theory: Assumes normal data fits within ellipsoid; detects outliers outside")


elliptic = EllipticEnvelope(contamination=0.05, random_state=42)
```

```
anomaly_results['Elliptic_Envelope'] = elliptic.fit_predict(X_scaled)

elliptic_scores = elliptic.score_samples(X_scaled)

anomaly_results['EE_Anomaly_Score'] = elliptic_scores

anomaly_results['EE_Anomaly_Probability'] = 1 / (1 + np.exp(elliptic_scores))


ee_anomalies = (anomaly_results['Elliptic_Envelope'] == -1).sum()

print(f"  ✓ Anomalies Detected: {ee_anomalies}
({ee_anomalies/len(anomaly_results)*100:.2f}%)")


print("\n[Algorithm 4] DBSCAN (Density-Based Clustering)")

print("  Parameters: eps=2.5, min_samples=5")

print("  Theory: Groups closely packed points; marks isolated points as anomalies")


dbscan = DBSCAN(eps=2.5, min_samples=5, n_jobs=-1)

anomaly_results['DBSCAN'] = np.where(dbscan.fit_predict(X_scaled) == -1, -1, 1)


dbscan_anomalies = (anomaly_results['DBSCAN'] == -1).sum()

print(f"  ✓ Anomalies Detected: {dbscan_anomalies}
({dbscan_anomalies/len(anomaly_results)*100:.2f}%)")


print("\n[Algorithm 5] Z-SCORE BASED DETECTION")

print("  Parameters: threshold=3 (standard deviations from mean)")

print("  Theory: Flags values that deviate significantly from mean")
```

```
z_scores = np.abs(zscore(X_scaled))

anomaly_results['Z_Score'] = np.where(np.any(z_scores > 3, axis=1), -1, 1)


z_score_anomalies = (anomaly_results['Z_Score'] == -1).sum()

print(f" ✓ Anomalies Detected: {z_score_anomalies}
({z_score_anomalies/len(anomaly_results)*100:.2f}%)")


print("\n[Algorithm 6] ENSEMBLE VOTING")

print("  Theory: Combines predictions from all methods for high-confidence detection")


anomaly_results['Ensemble_Votes'] = anomaly_results[['Isolation_Forest', 'LOF',
'Elliptic_Envelope', 'DBSCAN', 'Z_Score']].apply(lambda x: (x == -1).sum(), axis=1)

anomaly_results['Ensemble_Prediction_Strict'] =
np.where(anomaly_results['Ensemble_Votes'] >= 3, -1, 1)

anomaly_results['Ensemble_Prediction_Medium'] =
np.where(anomaly_results['Ensemble_Votes'] >= 2, -1, 1)


ensemble_anomalies_strict = (anomaly_results['Ensemble_Prediction_Strict'] == -1).sum()

ensemble_anomalies_medium = (anomaly_results['Ensemble_Prediction_Medium'] == -
1).sum()

print(f" ✓ High-Confidence Anomalies (3+ votes): {ensemble_anomalies_strict}
({ensemble_anomalies_strict/len(anomaly_results)*100:.2f}%)")

print(f" ✓ Medium-Confidence Anomalies (2+ votes): {ensemble_anomalies_medium}
({ensemble_anomalies_medium/len(anomaly_results)*100:.2f}%)")


print("\n[STEP 9] CREATING COMPREHENSIVE VISUALIZATIONS")
```

```
print("-" * 90)


fig1, axes1 = plt.subplots(2, 2, figsize=(16, 10))

fig1.suptitle('Anomaly Detection Results - Pricing Fraud Analysis', fontsize=16,
fontweight='bold')


ax1 = axes1[0, 0]

normal_prices = anomaly_results[anomaly_results['Ensemble_Prediction_Medium'] ==
1]['Original_Price']

anomaly_prices = anomaly_results[anomaly_results['Ensemble_Prediction_Medium'] == -
1]['Original_Price']


ax1.hist(normal_prices, bins=50, alpha=0.7, label='Normal', color='blue', edgecolor='black')

ax1.hist(anomaly_prices, bins=30, alpha=0.7, label='Anomalies', color='red',
edgecolor='black')

ax1.set_xlabel('Price ($)', fontsize=11)

ax1.set_ylabel('Frequency', fontsize=11)

ax1.set_title('Price Distribution: Normal vs Anomalies', fontsize=12, fontweight='bold')

ax1.legend(fontsize=10)

ax1.grid(True, alpha=0.3)


ax2 = axes1[0, 1]

scatter = ax2.scatter(anomaly_results['PCA1'], anomaly_results['PCA2'],

                      c=anomaly_results['Ensemble_Votes'], cmap='RdYlGn_r',

                      alpha=0.6, s=50, edgecolors='black', linewidth=0.5)
```

```
ax2.set_xlabel(f'PC1 ({pca.explained_variance_ratio_[0]*100:.1f}%)', fontsize=11)

ax2.set_ylabel(f'PC2 ({pca.explained_variance_ratio_[1]*100:.1f}%)', fontsize=11)

ax2.set_title('PCA Visualization - Anomaly Detection', fontsize=12, fontweight='bold')

cbar = plt.colorbar(scatter, ax=ax2)

cbar.set_label('Ensemble Votes', fontsize=10)

ax2.grid(True, alpha=0.3)


ax3 = axes1[1, 0]

detection_summary = pd.DataFrame({

   'Algorithm': ['Isolation\nForest', 'LOF', 'Elliptic\nEnvelope', 'DBSCAN', 'Z-Score',
'Ensemble\n(2+ votes)'],

   'Anomalies': [iso_anomalies, lof_anomalies, ee_anomalies, dbscan_anomalies,

                 z_score_anomalies, ensemble_anomalies_medium]

})

bars = ax3.bar(detection_summary['Algorithm'], detection_summary['Anomalies'],

               color=['skyblue', 'lightcoral', 'lightgreen', 'gold', 'plum', 'orange'],

               edgecolor='black', linewidth=1.5)

ax3.set_ylabel('Number of Anomalies', fontsize=11)

ax3.set_title('Algorithm Comparison', fontsize=12, fontweight='bold')

ax3.grid(True, alpha=0.3, axis='y')


for bar in bars:

   height = bar.get_height()
```

```
    ax3.text(bar.get_x() + bar.get_width()/2., height,
             f'{int(height)}', ha='center', va='bottom', fontsize=9, fontweight='bold')


ax4 = axes1[1, 1]
classification_counts = anomaly_results['Price_Classification'].value_counts()
colors_map = {'OVERPRICED': '#FF4444', 'UNDERPRICED': '#44FF44', 'NORMAL':
'#4444FF'}
colors = [colors_map.get(x, 'gray') for x in classification_counts.index]
bars4 = ax4.bar(classification_counts.index, classification_counts.values,
                color=colors, edgecolor='black', linewidth=1.5)
ax4.set_ylabel('Count', fontsize=11)
ax4.set_title('Price Classification Distribution', fontsize=12, fontweight='bold')
ax4.grid(True, alpha=0.3, axis='y')


for bar in bars4:
    height = bar.get_height()
    ax4.text(bar.get_x() + bar.get_width()/2., height,
             f'{int(height)}', ha='center', va='bottom', fontsize=10, fontweight='bold')


plt.tight_layout()
plt.show()


print("✓ Main visualization created")
```

```
fig2, ax = plt.subplots(figsize=(12, 6))
vote_dist = anomaly_results['Ensemble_Votes'].value_counts().sort_index()
ax.bar(vote_dist.index, vote_dist.values, color='steelblue', edgecolor='black', linewidth=1.5)
ax.set_xlabel('Number of Methods Detecting Anomaly', fontsize=12)
ax.set_ylabel('Number of Vehicles', fontsize=12)
ax.set_title('Ensemble Voting Distribution', fontsize=13, fontweight='bold')
ax.set_xticks([0, 1, 2, 3, 4, 5])
ax.grid(True, alpha=0.3, axis='y')
plt.tight_layout()
plt.show()

print("✓ Ensemble voting visualization created")

fig3, ax = plt.subplots(figsize=(14, 10))
correlation_matrix = X.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0,
            square=True, ax=ax, fmt='.2f', cbar_kws={'label': 'Correlation'},
            linewidths=0.5, linecolor='gray')
ax.set_title('Feature Correlation Matrix', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()
```

```
print("✓ Correlation heatmap created")


fig4, axes4 = plt.subplots(1, 2, figsize=(14, 6))


ax4_1 = axes4[0]

deviation_data = anomaly_results[anomaly_results['Ensemble_Prediction_Medium'] == -1]['Price_Deviation_Percent']

ax4_1.hist(deviation_data, bins=30, color='coral', edgecolor='black', alpha=0.7)

ax4_1.set_xlabel('Price Deviation (%)', fontsize=11)

ax4_1.set_ylabel('Frequency', fontsize=11)

ax4_1.set_title('Distribution of Price Deviations (Anomalies Only)', fontsize=12, fontweight='bold')

ax4_1.axvline(0, color='red', linestyle='--', linewidth=2, label='0% (Fair Price)')

ax4_1.legend()

ax4_1.grid(True, alpha=0.3)


ax4_2 = axes4[1]

classification_groups = anomaly_results[anomaly_results['Ensemble_Prediction_Medium'] == -1].groupby('Price_Classification')['Price_Deviation_Percent']

box_data = [group.values for name, group in classification_groups]

box_labels = [name for name, group in classification_groups]

ax4_2.boxplot(box_data, labels=box_labels)

ax4_2.set_ylabel('Price Deviation (%)', fontsize=11)

ax4_2.set_title('Price Deviation by Classification', fontsize=12, fontweight='bold')
```

```
ax4_2.grid(True, alpha=0.3, axis='y')


plt.tight_layout()

plt.show()


print("✓ Price deviation analysis created")


print("\n[STEP 11] STATISTICAL SUMMARY TABLES")

print("-" * 90)


summary_by_anomaly = pd.DataFrame({

    'Metric': ['Count', 'Avg Price', 'Avg Mileage', 'Avg Car Age', 'Avg Engine Size'],

    'Normal': [

        (anomaly_results['Ensemble_Prediction_Medium'] == 1).sum(),

        df_processed[anomaly_results['Ensemble_Prediction_Medium'] == 1]['Price'].mean(),

        df_processed[anomaly_results['Ensemble_Prediction_Medium'] ==
1]['Mileage'].mean(),

        df_processed[anomaly_results['Ensemble_Prediction_Medium'] ==
1]['Car_Age'].mean(),

        df_processed[anomaly_results['Ensemble_Prediction_Medium'] == 1]['Engine
size'].mean(),

    ],

    'Anomalies': [

        (anomaly_results['Ensemble_Prediction_Medium'] == -1).sum(),
```

```
        df_processed[anomaly_results['Ensemble_Prediction_Medium'] == -1]['Price'].mean(),
        df_processed[anomaly_results['Ensemble_Prediction_Medium'] == -1]['Mileage'].mean(),
        df_processed[anomaly_results['Ensemble_Prediction_Medium'] == -1]['Car_Age'].mean(),
        df_processed[anomaly_results['Ensemble_Prediction_Medium'] == -1]['Engine size'].mean(),
    ]
})

print("\nComparison: Normal vs Anomalous Vehicles")
print(summary_by_anomaly.to_string(index=False))

print("\n\nAlgorithm Agreement Analysis:")
agreement_matrix = pd.crosstab(
    anomaly_results['Ensemble_Votes'],
    anomaly_results['Price_Classification'],
    margins=True
)
print(agreement_matrix)

print("\n[STEP 12] FEATURE IMPORTANCE ANALYSIS")
print("-" * 90)
```

```
feature_anomaly_corr = pd.DataFrame({
    'Feature': feature_columns,
    'Corr_with_IF_Score': [X_scaled_df[col].corr(anomaly_results['IF_Anomaly_Score']) for col in feature_columns],
    'Corr_with_LOF_Score': [X_scaled_df[col].corr(anomaly_results['LOF_Score']) for col in feature_columns],
    'Avg_Abs_Correlation': [
        (abs(X_scaled_df[col].corr(anomaly_results['IF_Anomaly_Score'])) +
        abs(X_scaled_df[col].corr(anomaly_results['LOF_Score']))) / 2
        for col in feature_columns
    ]
}).sort_values('Avg_Abs_Correlation', ascending=False)

print("\nTop Features Contributing to Anomaly Detection:")
print(feature_anomaly_corr.head(8).to_string(index=False))

print("\n" + "=" * 90)
print("COMPLETE PROJECT EXECUTION FINISHED!")
print("=" * 90)
print(f"\nTotal Processing Time: Check console for execution time")
print(f"All output files saved successfully!")
print(f"\nKey Metrics Summary:")
print(f"  • Total Vehicles Analyzed: {len(df_processed):,}")
print(f"  • High-Confidence Anomalies: {ensemble_anomalies_strict}")
```

```
print(f"  • Medium-Confidence Anomalies: {ensemble_anomalies_medium}")

print(f"  • Overpriced Vehicles Detected: {(anomaly_results['Price_Classification'] == 'OVERPRICED').sum()}")

print(f"  • Underpriced Vehicles Detected: {(anomaly_results['Price_Classification'] == 'UNDERPRICED').sum()}")

print(f"\nRecommended Action: Review 'high_confidence_anomalies.csv' immediately!")
```