**Machine Learning Algorithms in SmartFleet Route Optimizer**

The app integrates multiple ML algorithms to optimize routes, reduce costs, and enhance safety using data from crime, weather, traffic, sensors, and news sentiment. Below is a comprehensive list of the ML algorithms based on the application.py code and their contributions:

1. **Crime Prediction Model (Statistical Aggregation)**:
   * **Description**: A statistical model that predicts the probability of violent crimes (e.g., assault, battery, robbery, homicide, criminal sexual assault) across Chicago zones and hours.
   * **Code Reference** (in preprocess\_data):

python

violent\_types = ['ASSAULT', 'BATTERY', 'ROBBERY', 'HOMICIDE', 'CRIMINAL SEXUAL ASSAULT']

crime\_df\_local = crime\_df\_local[crime\_df\_local['Primary Type'].isin(violent\_types)]

crime\_df\_local['zone'] = crime\_df\_local['Community Area'].fillna(1).astype(int).clip(1, 77)

crime\_counts = crime\_df\_local.groupby(['zone', 'hour']).size().reset\_index(name='count')

total\_per\_zone = crime\_counts.groupby('zone')['count'].transform('sum')

crime\_risk = crime\_counts.assign(crime\_prob=crime\_counts['count'] / total\_per\_zone)

crime\_risk = crime\_risk.pivot(index='zone', columns='hour', values='crime\_prob').fillna(0).reset\_index()

* + **Algorithm Details**:
    - **Type**: Statistical aggregation (frequency-based probability estimation).
    - **Input**: Chicago crime data (Chicago Crime Sampled.csv) with timestamps and community areas.
    - **Output**: crime\_prob (probability of violent crime per zone and hour).
    - **Role**: Contributes to total\_risk in calculate\_route\_metrics, increasing risk\_avoided for high-crime zones, which drives **Crime Avoidance** savings (20% of annual\_savings) in the **What-If Cost Estimator**.
  + **Impact**: Enables safer routes by avoiding high-crime zones, reducing incident-related costs (e.g., theft, damage).

1. **Weather Risk Model (Rule-Based Classifier)**:
   * **Description**: A rule-based model that assesses weather-related risks based on temperature and precipitation thresholds.
   * **Code Reference** (in preprocess\_data):

python

weather\_df\_local['temp\_f'] = weather\_df\_local.get('TMAX', 0) / 10

weather\_df\_local['precip\_in'] = weather\_df\_local.get('PRCP', 0) / 10

weather\_df\_local['weather\_risk'] = ((weather\_df\_local['temp\_f'] > 80) | (weather\_df\_local['precip\_in'] > 0.1)).astype(float) \* 2

weather\_risk\_hourly = weather\_df\_local.groupby('hour')['weather\_risk'].mean().reset\_index()

* + **Algorithm Details**:
    - **Type**: Rule-based classifier (threshold-based decision rules).
    - **Input**: Weather data (Chicago Weather.csv) with temperature (TMAX) and precipitation (PRCP).
    - **Output**: weather\_risk (0 or 2, based on high temperature > 80°F or precipitation > 0.1 inches).
    - **Role**: Contributes to total\_risk in calculate\_route\_metrics, scaled by weather\_factor (based on precipitation, wind, visibility), influencing risk\_avoided. Indirectly affects **Fuel Efficiency** savings (20% of annual\_savings) by optimizing routes around adverse weather.
  + **Impact**: Enhances route safety and efficiency by avoiding weather-related risks, reducing delays and fuel costs.

1. **Sentiment Analysis LLM (Natural Language Processing)**:
   * **Description**: A large language model (LLM) that analyzes news sentiment to identify negative events (e.g., protests, unrest) that increase route risk.
   * **Code Reference** (in preprocess\_data):

python

sentiment\_df\_local = sentiment\_df.copy()

sentiment\_df\_local['date'] = pd.to\_datetime(sentiment\_df\_local['date'], errors='coerce')

data = data.merge(sentiment\_df\_local, left\_on='pickup\_time', right\_on='date', how='left')

data['sentiment\_risk'] = data['sentiment\_score'].fillna(0).apply(lambda x: 1.0 if x < -0.5 else 0)

data['total\_risk'] += data['sentiment\_risk']

* + **Algorithm Details**:
    - **Type**: Transformer-based LLM (assumed, as sentiment\_score is precomputed in News Sentiment Analysis.csv).
    - **Input**: News data with sentiment\_score (-1 to 1, likely from an LLM like BERT or RoBERTa).
    - **Output**: sentiment\_risk (1.0 for negative sentiment < -0.5, else 0).
    - **Role**: Contributes to total\_risk in calculate\_route\_metrics, increasing risk\_avoided for zones with negative sentiment. Drives **Sentiment-Based Avoidance** savings (10% of annual\_savings) in the **What-If Cost Estimator**.
  + **Impact**: Reduces risks from events reported in news (e.g., protests), lowering incident-related costs.

1. **Sensor Degradation Model (Quadratic Regression)**:
   * **Description**: A model predicting vision sensor accuracy degradation based on humidity, used for maintenance scheduling.
   * **Code Reference** (in create\_vision\_graph and calculate\_route\_metrics):

python

humidity\_range = np.linspace(0, 100, 100)

base\_accuracy = 0.95

failure\_rate = 0.0008 \* (humidity\_range - 40)\*\*2

vision\_accuracy = base\_accuracy \* (1 - failure\_rate)

* + **Algorithm Details**:
    - **Type**: Quadratic regression (parametric model for failure rate).
    - **Input**: Humidity percentage (user input or data-driven).
    - **Output**: vision\_accuracy (sensor accuracy as a percentage, reduced by humidity-induced failure rate).
    - **Role**: Influences sensor\_penalty in calculate\_route\_metrics (sensor\_penalty = (1 - vision\_accuracy.mean()) \* 500), affecting total\_reward. Drives **Sensor Maintenance** savings (40% of annual\_savings, capped at $27,000) by predicting optimal replacement times.
  + **Impact**: Reduces downtime and repair costs by scheduling early sensor replacements, reflected in the **Sensor Maintenance** savings.

1. **Composite Risk Model (Weighted Ensemble)**:
   * **Description**: An ensemble model combining crime, weather, sentiment, traffic, and sensor risks to compute a total risk score for route optimization.
   * **Code Reference** (in calculate\_route\_metrics):

python

crime\_risk\_norm = min(row['crime\_prob'] / 0.05, 1.0) if row['crime\_prob'] > 0 else 0

weather\_risk\_norm = min(row['weather\_risk'] / 2.0, 1.0)

sensor\_risk\_norm = min(total\_sensor\_risk.iloc[\_ % len(total\_sensor\_risk)] / 2.0, 1.0)

sentiment\_risk\_norm = min(row['sentiment\_risk'], 1.0)

traffic\_risk\_norm = min(row['traffic\_risk'], 1.0)

total\_risk = (crime\_risk\_norm + weather\_risk\_norm + sensor\_risk\_norm +

sentiment\_risk\_norm + traffic\_risk\_norm) \* weather\_factor \* temporal\_factor / 5.0

risk\_threshold = crime\_threshold \* (1 + maintenance\_impact \* 0.5) \* (vision\_weight + gyro\_weight) / 2.0

if total\_risk > risk\_threshold:

risk\_avoided += 1

* + **Algorithm Details**:
    - **Type**: Weighted ensemble (normalized sum of individual risk models, scaled by environmental and temporal factors).
    - **Input**: Outputs from crime model (crime\_prob), weather model (weather\_risk), sentiment LLM (sentiment\_risk), sensor model (total\_sensor\_risk), and traffic data (traffic\_risk).
    - **Output**: total\_risk (normalized risk score) and risk\_avoided (count of avoided high-risk zones).
    - **Role**: Drives route optimization by selecting safer routes, reflected in **Risk Avoided** (40-80%) and cost savings (crime, sentiment, fuel efficiency) in the **What-If Cost Estimator**.
  + **Impact**: Integrates all ML models to optimize routes, reducing costs and risks across multiple dimensions.

**Summary of ML Algorithms**

* **1. Crime Prediction Model**: Statistical aggregation for violent crime probabilities by zone and hour.
* **2. Weather Risk Model**: Rule-based classifier for weather risks based on temperature and precipitation.
* **3. Sentiment Analysis LLM**: Transformer-based LLM for news sentiment, identifying negative events.
* **4. Sensor Degradation Model**: Quadratic regression for vision sensor accuracy based on humidity.
* **5. Composite Risk Model**: Weighted ensemble combining all risks for route optimization.