Data Preparation & Ethical Data Handling

AI Masters Capstone Project - Presentation 2

Jonathan Agustin

November 2024

What We'll Cover Today

- Ethical data handling: fundamentals and importance
- Automated preprocessing: cleaning, transforming, and validating data
- Privacy & compliance: embedding regulations into data pipelines
- Creating effective validation sets (Thomas, 2017): going beyond random splits
- Detecting & mitigating bias before training

We'll bridge theory and practice, ensuring both technical rigor and ethical integrity.

Why Ethical Data Handling Matters

- Data = real people's lives, rights, and opportunities
- Trust & credibility: a long-term asset for AI-driven products
- Avoiding bias, protecting privacy, and ensuring fairness is both a moral and practical imperative

Ethical principles aren't just guidelines; they set the foundation for sustainable ML solutions.

Automated Data Preprocessing

- Detect & handle missing values, outliers, and inconsistencies programmatically
- Transform features uniformly: encoding categories, normalizing units
- Ingrain reproducibility and transparency into your data workflows

Automation reduces grunt work and lets you focus on building higher-quality, fairer models.

Practical Preprocessing Techniques

Load raw data df = pd.read$_csv$("$raw_data.csv$")

Impute missing numeric values numeric$_cols = df.select_dtypes(include = ['float','int']).columnsimputer = SimpleImputer(strategy ='mean')df[numeric_cols] = imputer.fit_transform(df[numeric_cols])$

One-hot encode categorical variables
categorical$_cols = df.select_dtypes(include = ['object']).columnsencoder = OneHotEncoder(sparse_output = False, handle_unknown =' ignore')encoded = encoder.fit_transform(df[categorical_cols])encoded_df = pd.DataFrame(encoded, columns = encoder.get_feature_names_out(categorical_cols))df = pd.concat([df.drop(columns = categorical_cols), encoded_df], axis = 1)$

Scale numeric features scaler = StandardScaler()
df[numeric$_cols] = scaler.fit_transform(df[numeric_cols])$

Data Quality Assurance

class InputSchema(pa.SchemaModel): age: Series[int] = pa.Field(ge=0, le=120) income: Series[float] = pa.Field(ge=0) Add more fields and constraints as needed class Config: strict = True

try: validated$_df = InputSchema.validate(df) except pa.errors.SchemaError as e: print("Data validation failed:", e) Halt pipeline, alert team$

Ensuring data integrity early prevents garbage-in, garbage-out scenarios.

Creating a Good Validation Set (Thomas, 2017)

- Validation set: used to choose models and tune parameters, not just to optimize the training score.
- Must mimic future data conditions to prevent over-optimistic performance estimates.
- Go beyond simple random splits; consider time-series constraints, new users, or new categories.

"A poorly chosen validation set can be the most important culprit for model failures in production" (Thomas, 2017).

When is a Random Subset Not Good Enough? (Thomas, 2017)

- **Time Series**: Validate on later time segments, not random points.
- **New Users or Entities**: If production data includes new people or new items, ensure your validation also excludes them from the training set.
- **Domain Shifts**: If the environment or context changes over time, replicate that change in your validation split.

This approach better simulates real-world deployment conditions.

Practical Examples (Thomas, 2017)

- **Time series (Kaggle grocery sales):** - Training: Jan 2013 to July 31, 2017 - Validation: Aug 1 to Aug 15, 2017 - Future (test) data: Aug 16 to Aug 31, 2017
- **New people or items (Distracted driver detection):** - Training: selected drivers - Validation: entirely different drivers This avoids overfitting to known individuals' characteristics.

These techniques reduce the gap between validation and real-world performance.

Kaggle Considerations (Thomas, 2017)

- Kaggle's "training data" = your training + validation sets
- The Kaggle test set is divided (unknown to you) into public and private parts
- A well-chosen validation set prevents overfitting to the public leaderboard and provides a truer measure of real-world performance.

Good validation strategies translate beyond competitions, ensuring robust, trustworthy ML solutions.

Privacy Protection and Compliance

if 'user$_id'$ $indf.columns : df['user_id_hashed'] = df['user_id'].apply(lambdax : hashlib.sha256(str(x).encode()).hexdigest())df.drop(columns = ['user_id'], inplace = True)$

Remove personally identifiable info pii$_c$ $ols = ['name','email','phone_number']df = df.drop(columns = [colforcolinpii_colsifcolindf.columns])$

By integrating privacy safeguards early, we show respect for users and uphold legal standards.

Detecting and Mitigating Bias

$\text{data} = \text{BinaryLabelDataset(}\ \text{favorable}_{l}abel = 1, unfavorable_{l}abel = 0, df = df, label_{n}ames = [\prime approved_{l}oan\prime], protected_{a}ttribute_{n}ames = [\prime gender\prime])$

$\text{metric} = \text{BinaryLabelDatasetMetric(}\ \text{data},$
$\text{unprivileged}_{g}roups = [\prime gender\prime : 0], privileged_{g}roups = [\prime gender\prime : 1])disparate_{i}mpact = metric.disparate_{i}mpact()if disparate_{i}mpact < 0.8 : rw = Reweighing(unprivileged_{g}roups = [\prime gender\prime : 0], privileged_{g}roups = [\prime gender\prime : 1])data_{b}alanced = rw.fit_{t}ransform(data)$

Catching bias early prevents amplifying injustices in production.

A Holistic Data Strategy

- Integrate cleaning, validation, privacy, bias mitigation, and careful validation set construction into one pipeline.
- Document all steps for transparency and reproducibility.
- Move forward with confidence that your data handling sets the stage for fair, accurate ML models.

These practices ensure your ML models are ready to face real-world conditions responsibly.

Next Steps

- Next Presentation: Automating Model Training & Ethical Evaluation
- Connect today's data pipeline principles to fair and robust model training

From data preparation to model training—the journey continues with ethics at the core.

References

- Thomas, R. (2017). *How (and why) to create a good validation set*. Retrieved from https://rachel.fast.ai/posts/2017-11-13-validation-sets/
- *aif360* toolkit: https://github.com/Trusted-AI/AIF360
- *pandera* library: https://pandera.readthedocs.io/
- GDPR guidelines: https://gdpr.eu/