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): beyond naive splits
- Detecting & mitigating dataset bias before model training

Our goal: robust, transparent, and ethical data pipelines that set the stage for trustworthy ML.

Why Ethical Data Handling Matters

- Data represents real human stories, with tangible impacts
- Trust & credibility: essential assets for long-term AI adoption
- Preventing bias, respecting privacy, and ensuring fairness safeguards users and protects organizations

Ethical standards in data handling underpin the entire ML lifecycle.

Automated Data Preprocessing

- Standardize cleaning: detect and handle missing values, outliers, inconsistencies
- Consistent transformations: one-hot encoding, scaling, normalization are reproducible
- Transparent pipelines: every step documented, reducing human error and hidden bias

Automation turns data wrangling into a reliable, transparent foundation for fair ML.

Practical Preprocessing Techniques

df = pd.read$_csv(”raw_data.csv”)$

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

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 additional constraints as needed class Config: strict = True

try: validated$_d f = InputSchema.validate(df) except pa.errors.SchemaError as e:$ $print("Data validation failed:", e) Halt pipeline, send alerts$

Automated checks ensure data integrity at every step, building trust in your pipeline.

Creating a Good Validation Set (Thomas, 2017)

- Validation = key to honest model assessment
- Must simulate future data conditions, not just be a random sample
- Consider time splits, new user groups, and domain shifts to reflect true deployment scenarios

As Thomas (2017) notes, poor validation sets often cause failure when models meet the real world.

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

- **Time Series**: Validate on data from a later time window
- **New Entities**: If future data includes new customers not seen before, validation should too
- **Domain Shifts**: Reflect changes in user behavior or market conditions in your validation strategy

Adopting these strategies reduces the risk of costly surprises post-deployment.

Practical Examples (Thomas, 2017)

- **Time Series (e.g., grocery sales)**: - Training: Jan 2013 - July 31, 2017 - Validation: Aug 1 - Aug 15, 2017 - Test: Aug 16 onward
- **New Users (e.g., distracted driver)**: - Training: certain known drivers - Validation: entirely new, unseen drivers

This tailored approach ensures validation metrics align with real-world performance.

Kaggle Considerations (Thomas, 2017)

- Kaggle's public leaderboard is a partial view; don't overfit to it
- Your custom validation set should approximate true future data, not just mimic the public leaderboard
- This ensures real, lasting performance gains rather than superficial improvements

A sound validation strategy is an investment in long-term model reliability.

Privacy Protection and Compliance

Pseudonymize user IDs if 'user$_i d' in df.columns : df['user_i d_h ashed'] = df['user_i d'].apply(lambda x : hashlib.sha256(str(x).encode()).hexdigest()) df.drop(columns = ['user_i d'], inplace = True)$

Remove PII (Personally Identifiable Information) pii$_c ols = ['name', 'email', 'phone_n umber'] df = df.drop(columns = [col for col in pii_c ols if col in df.columns])$

Privacy-by-design: integrate compliance and respect for users at the data level.

Detecting and Mitigating Bias

data = BinaryLabelDataset(favorable$_label = 1, unfavorable_label = 0, df = df, label_names = ['approved_loan'], protected_attribute_names = ['gender'])$

metric = BinaryLabelDatasetMetric(data, unprivileged$_groups = ['gender': 0], privileged_groups = ['gender': 1])$

if metric.disparate$_impact() < 0.8 : rw = Reweighing(unprivileged_groups = ['gender': 0], privileged_groups = ['gender': 1]) data_balanced = rw.fit_transform(data)$

Embedding fairness checks ensures our ML solutions serve society responsibly.

A Holistic Data Strategy

- Integrate cleaning, validation, privacy, bias mitigation, and realistic validation strategies
- Document processes for transparency and auditability
- Lay a foundation that upholds fairness, compliance, and trust through the entire ML pipeline

Such an approach is not only ethically sound but also yields more robust, future-proof models.

Next Steps

- Next Presentation: Automating Model Training & Ethical Evaluation
- Build upon today's principles to ensure model choices and hyperparameter tuning respect the same ethical and rigorous standards

Our pipeline is now primed for fair and responsible model development.

References

- Thomas, R. (2017). *How (and why) to create a good validation set*. 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/