Building and Deploying Fair and Unbiased Machine Learning Systems: An Art, Not Science

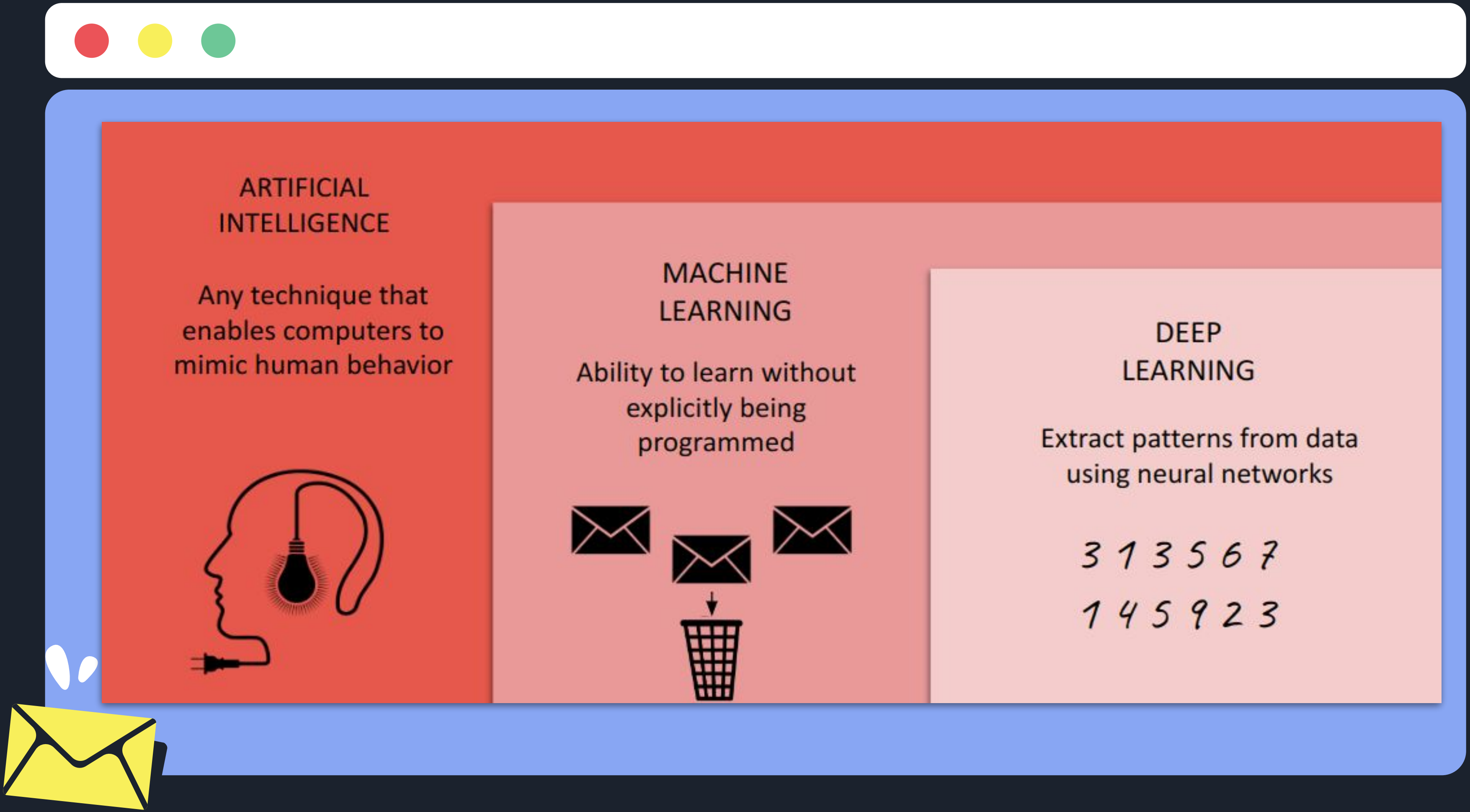
By: Rashmi Nagpal

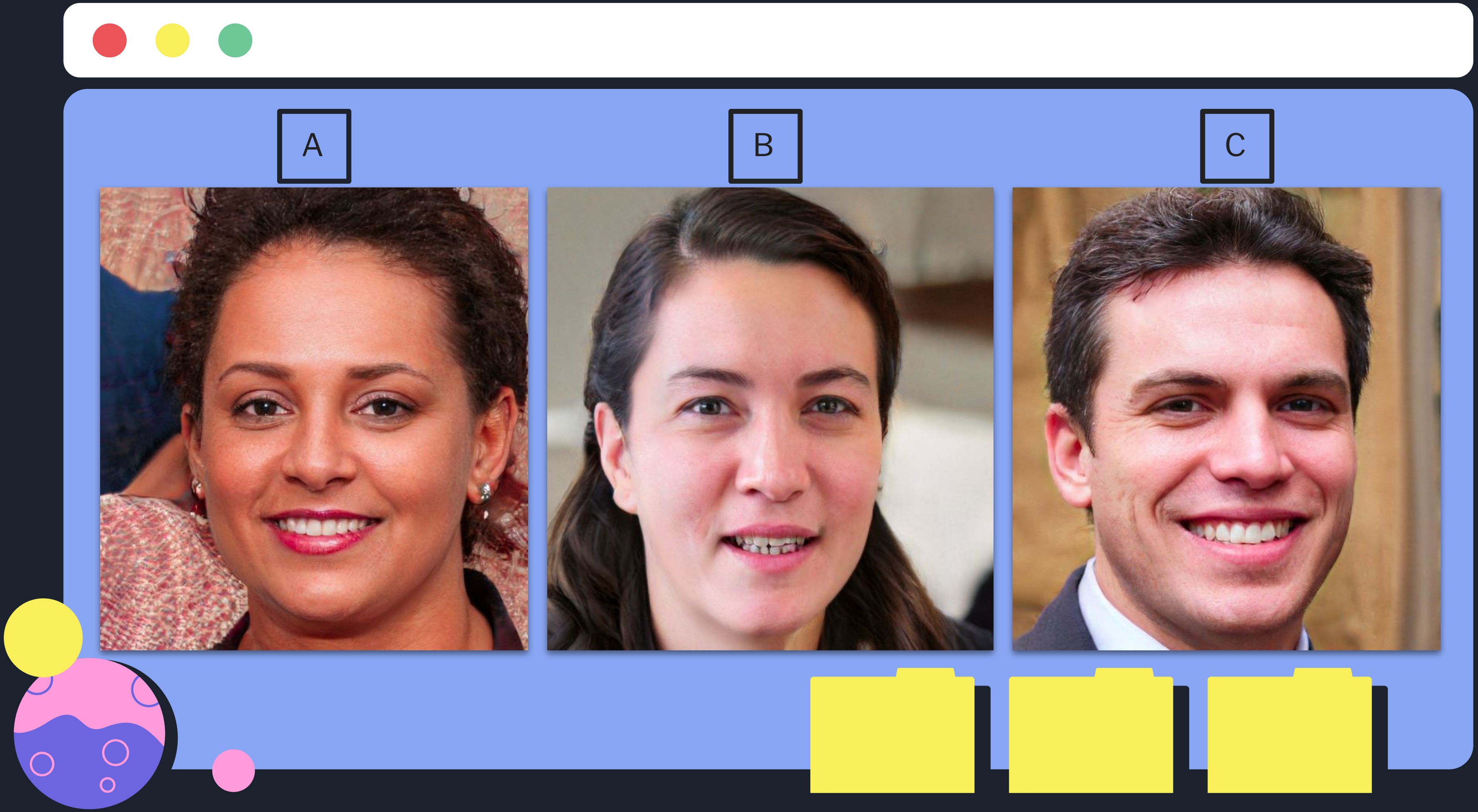
Agenda

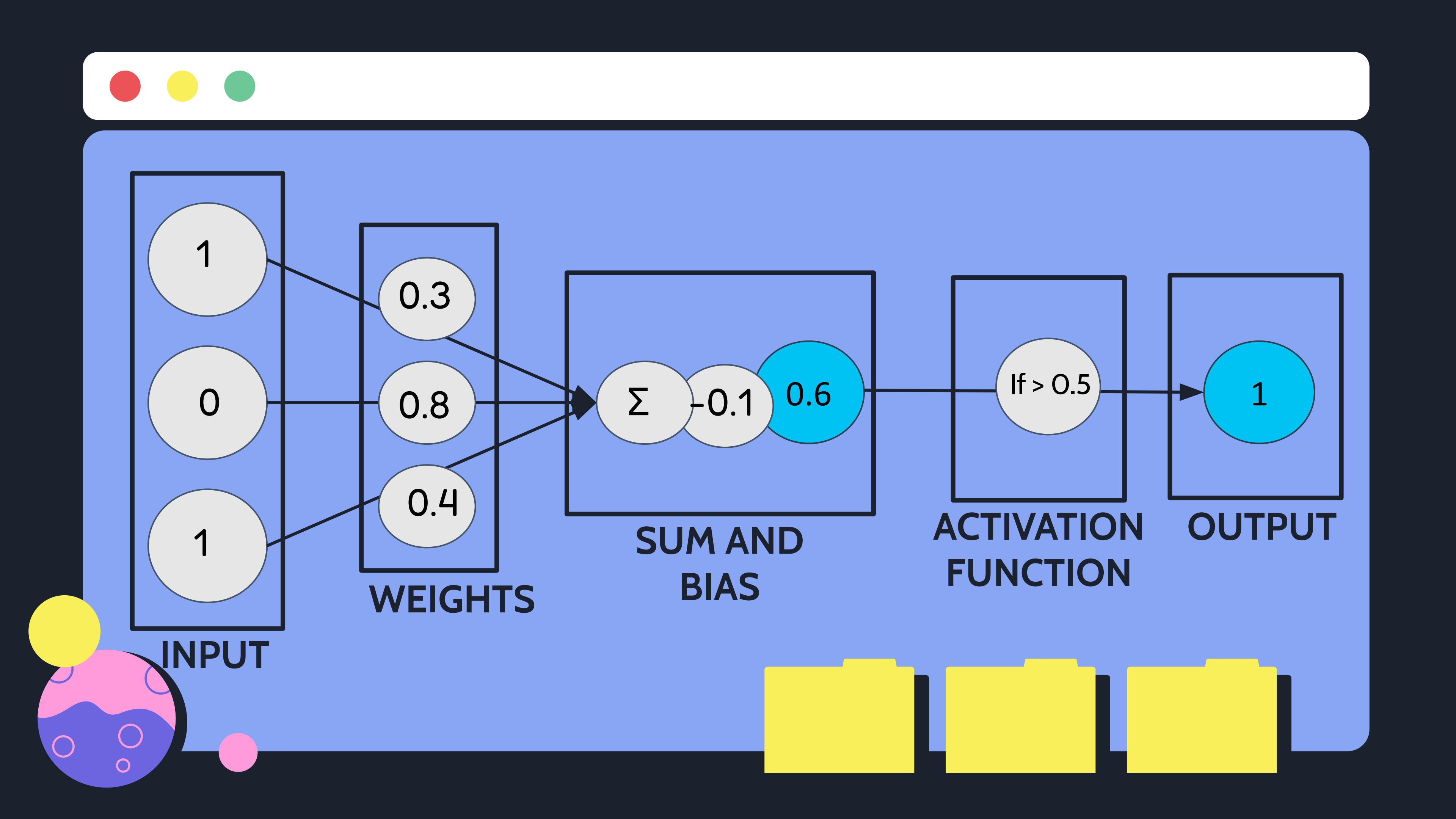
Introduction	What is machine learning and its related concepts!
Understanding Black-Box	How machine learning models can lead to unfair, biased decisions
Building fair & unbiased models	What are the strategies to build fair and unbiased machine learning models
ML Test Pyramid	What are the levels of testing in ML Test Pyramid
Technical Debts in ML Sys	What are the strategies to address technical debts
Conclusion	What are the key-takeaways from this talk!

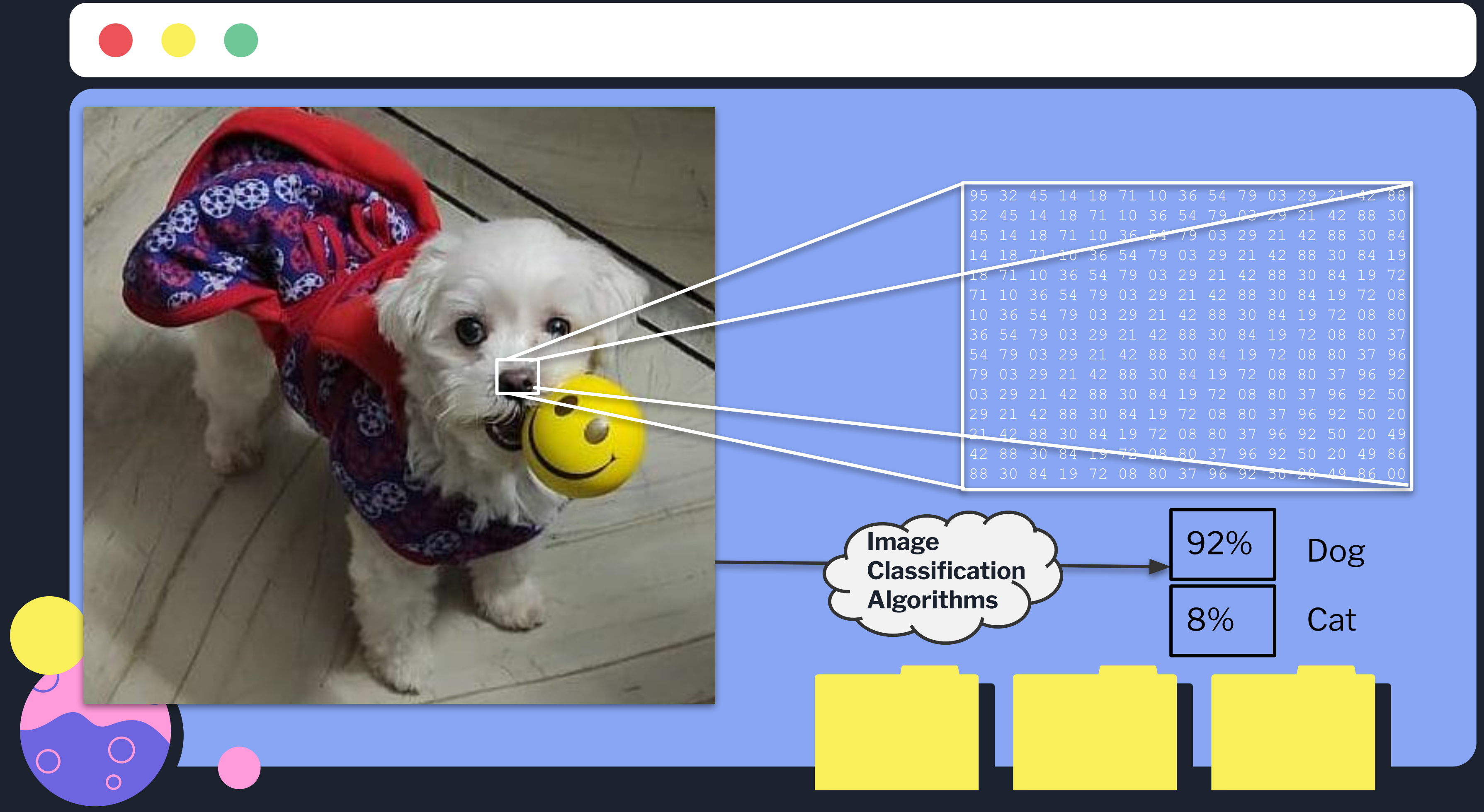
01

Introduction

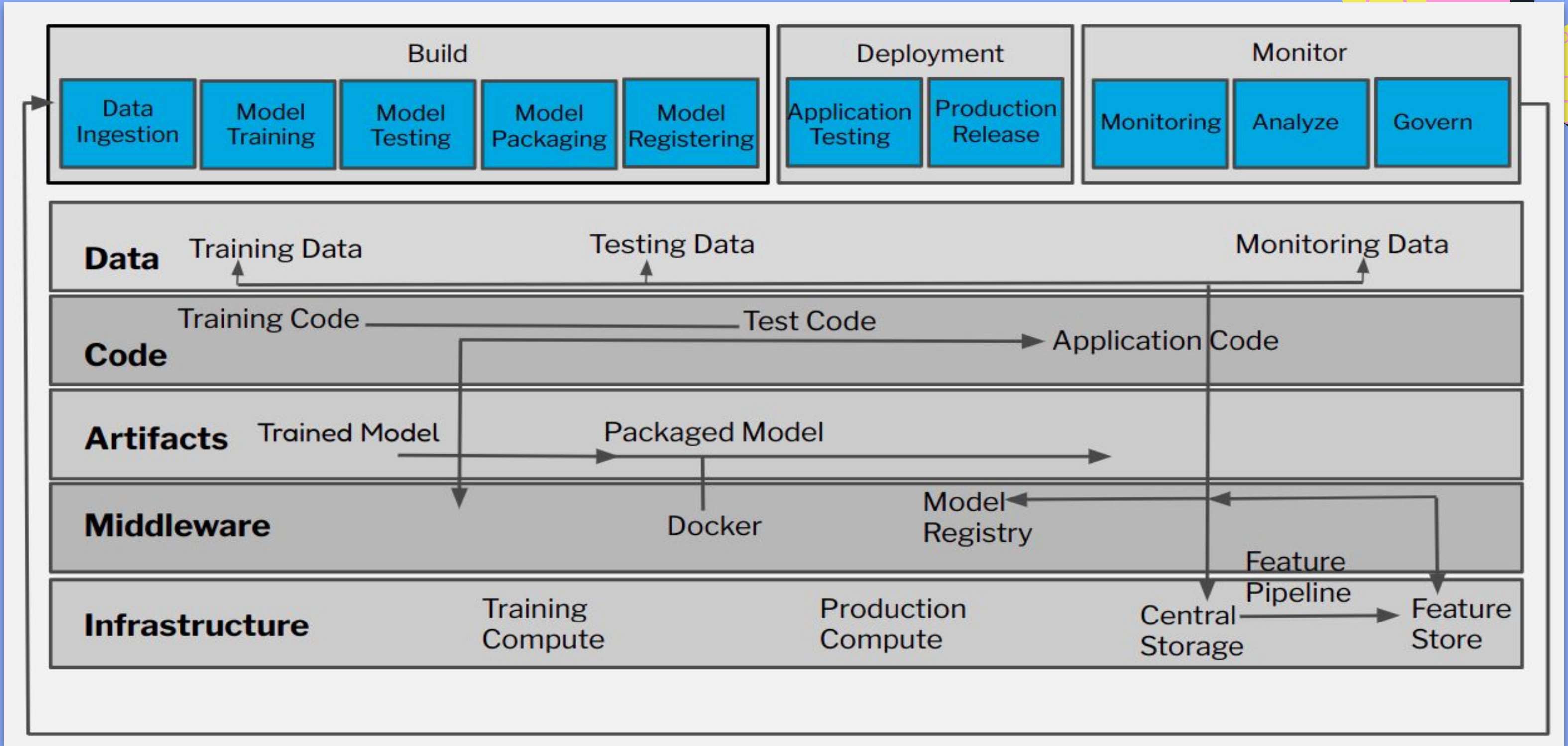

ARTIFICIAL INTELLIGENCE
Any technique that enables computers to mimic human behavior
MACHINE LEARNING
Ability to learn without explicitly being programmed
DEEP LEARNING
Extract patterns from data using neural networks
3 1 3 5 6 7
1 4 5 9 2 3


A
B
C


1
0
1
INPUT
0.3
0.8
0.4
WEIGHTS
Σ
-0.1
0.6
SUM AND BIAS
If > 0.5
ACTIVATION FUNCTION
1
OUTPUT



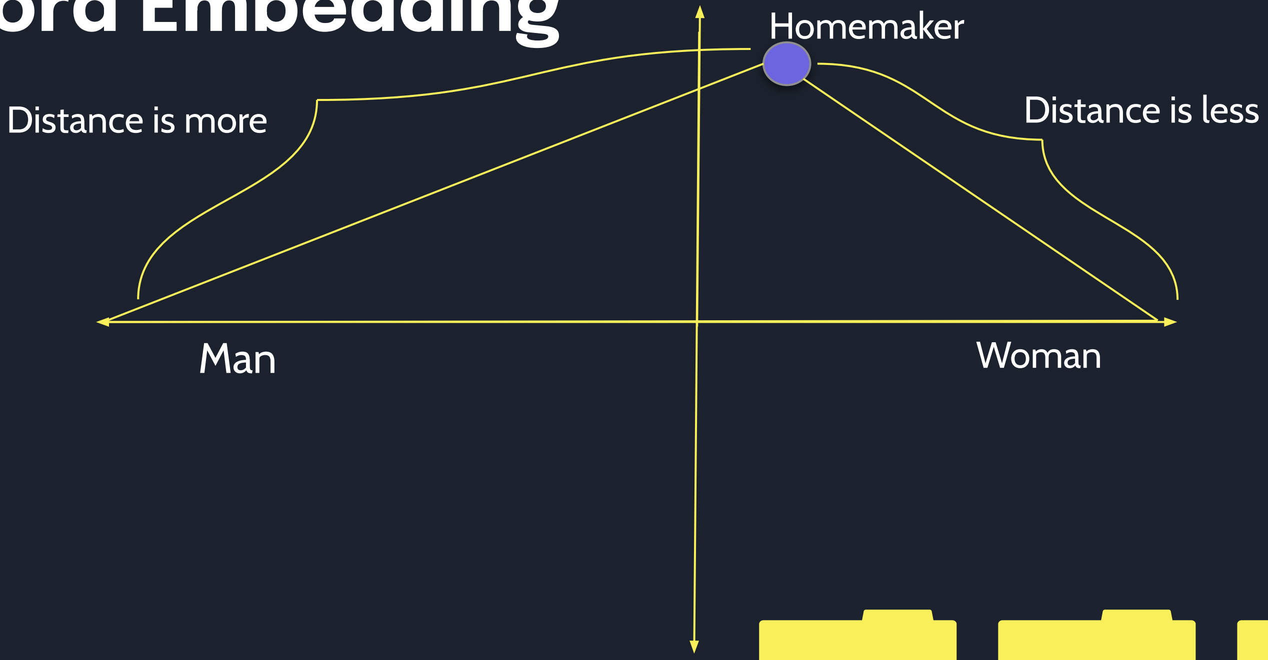
95 32 45 14 18 71 10 36 54 79 03 29 21 42 88
32 45 14 18 71 10 36 54 79 03 29 21 42 88 30
45 14 18 71 10 36 54 79 03 29 21 42 88 30 84
14 18 71 10 36 54 79 03 29 21 42 88 30 84 19
18 71 10 36 54 79 03 29 21 42 88 30 84 19 72
71 10 36 54 79 03 29 21 42 88 30 84 19 72 08
10 36 54 79 03 29 21 42 88 30 84 19 72 08 80
36 54 79 03 29 21 42 88 30 84 19 72 08 80 37
54 79 03 29 21 42 88 30 84 19 72 08 80 37 96
79 03 29 21 42 88 30 84 19 72 08 80 37 96 92
03 29 21 42 88 30 84 19 72 08 80 37 96 92 50
29 21 42 88 30 84 19 72 08 80 37 96 92 50 20
21 42 88 30 84 19 72 08 80 37 96 92 50 20 49
42 88 30 84 19 72 08 80 37 96 92 50 20 49 86
88 30 84 19 72 08 80 37 96 92 50 20 49 86 00
Image Classification Algorithms
92%
Dog
8%
Cat


Build
Data Ingestion
Model Training
Model Testing
Model Packaging
Model Registering
Deployment
Application Testing
Production Release
Monitor
Monitoring
Analyze
Govern
Data
Training Data
Testing Data
Monitoring Data
Code
Training Code
Test Code
Application Code
Artifacts
Trained Model
Packaged Model
Middleware
Docker
Model Registry
Feature Pipeline
Infrastructure
Training Compute
Production Compute
Central Storage
Feature Store

02

Black-Box Problem

Word Embedding



ChatGPT

The doctor yelled at the nurse because he was late. Who was late?

In the sentence "The doctor yelled at the nurse because he was late," the pronoun "he" refers back to "the doctor." Therefore, the doctor was the one who was late.

The doctor apologized to the nurse for her late arrival to the emergency room. Who was late previously?

Based on the sentence you provided, it seems that the doctor was the one who was late to the emergency room, and that's why he apologized to the nurse.

ChatGPT

The nurse apologized to the doctor for his late arrival to the operation room. Who was late?

Based on the sentence you provided, it seems that there is an error in the sentence construction that could cause confusion about who was actually late. The use of pronouns in this sentence makes it unclear who was late to the operation room. However, if we assume that the sentence was intended to mean that the nurse was apologizing to the doctor for her own late arrival to the operation room, then the nurse was the one who was late.

Credit Score

03

Why biased, unfair output?

01 Biased training dataset

02 Cognitive bias

03 Lack of diversity within dataset

04 Inadequate evaluation metrics

04

How to build unbiased, fair ML models?

01 **Diverse Data**

Collect well representative dataset

02 **Pre-process Data Pipeline**

Use data augmentation, feature selection techniques.

03 **Fairness Techniques**

Use algorithmic fairness measures like equalized odds, statistical parity et al.

04 **Testing end-to-end ML pipeline**

Add model testing, unit testing and robustness testing techniques

05 **Use Explainability frameworks**

Use framework like LIME , SHAP

```
# Load the data
data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data',  header=None,sep=' ')

# Define feature names
feature_names = ['status', 'duration', 'credit_history', 'purpose', 'amount', 'savings', 'employment_duration', 'installment_rate', 'statussex',
'other_debtors', 'residence_since', 'property', 'age', 'other_installment_plans', 'housing', 'number_credits', 'job', 'people_liable',
'telephone', 'foreign_worker', 'labels']

# Assign the feature names to the dataset
data.columns = feature_names

# Convert categorical variables to numerical using one-hot encoding
categorical_cols = ['status', 'credit_history', 'purpose', 'savings', 'employment_duration',  'statussex', 'other_debtors', 'property',
'other_installment_plans', 'housing', 'job', 'telephone', 'foreign_worker']
data = pd.get_dummies(data, columns=categorical_cols)

# Define the target variable and the features
target_col = 'labels'

features = data.drop(target_col, axis=1)

target = data[target_col]
```

```python
# Apply SMOTE : Synthetic Minority Oversampling Technique (SMOTE) is a statistical technique for increasing the number of cases in your dataset in a balanced way.
sm = SMOTE(sampling_strategy='auto')
X_resampled, y_resampled = sm.fit_resample(features , target)

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)

print("Before applying SMOTE technique \n")
unique, counts = np.unique(y_train, return_counts=True)
print(dict(zip(unique, counts)))

print("\n After applying SMOTE technique \n")
unique, counts = np.unique(y_resampled, return_counts=True)
print(dict(zip(unique, counts)))
```

```
Before applying SMOTE technique

{0: 569, 1: 551}

After applying SMOTE technique

{0: 700, 1: 700}
```

```python
# Define the XGBoost model
params = {
  'objective': 'binary:logistic',
  'eval_metric': 'auc',
  'tree_method': 'hist',
  'max_depth': 8,
  'learning_rate': 0.1,
  'n_estimators': 100,
  'seed': 42
}

xg_model = xgb.XGBClassifier(**params)

# Train the model
xg_model.fit(X_train_balanced, y_train_balanced)
```

```python
# Predict the test set
xgboost_y_pred = xg_model.predict(X_test)

# Evaluate the model
print(f'Accuracy: {accuracy_score(y_test, xgboost_y_pred):.4f}')
print(classification_report(y_test, xgboost_y_pred))

# Get the predicted labels for the privileged and unprivileged groups
privileged = y_pred[(X_test['statussex_A91'] == 1) | (X_test['statussex_A93'] == 1) | (X_test['statussex_A94'] == 1) & (X_test['age'] > 18)]
unprivileged = y_pred[(X_test['statussex_A92'] == 1) & (X_test['age'] > 18)]
```

```
Accuracy: 0.8200
              precision    recall  f1-score   support

           0       0.72      0.64      0.68        59
           1       0.86      0.89      0.88       141

    accuracy                           0.82       200
   macro avg       0.79      0.77      0.78       200
weighted avg       0.82      0.82      0.82       200

              Score
Precision  0.857143
Recall     0.893617
F1-Score   0.875000
Accuracy   0.820000
```

```python
# Calculate the proportions of positive predictions for each group
privileged_proportion = np.mean(privileged)
unprivileged_proportion = np.mean(unprivileged)

# Define the search space
min_bound = [0.001, 1, 10]
max_bound = [0.5, 10, 200]
bounds = (min_bound, max_bound)

# Initialize the particle swarm
options = {'c1': 0.5, 'c2': 0.3, 'w': 0.9}
num_particles = 2
dimensions = len(min_bound)
optimizer = ps.single.GlobalBestPSO(n_particles=num_particles, dimensions=dimensions, options=options, bounds=bounds)
```

```python
# Run the optimization
cost, pos = optimizer.optimize(objective_function, iters=10)

# Calculate the Statistical Parity Difference
statistical_parity_difference = privileged_proportion - unprivileged_proportion

# Print the Statistical Parity Difference
print(f"Statistical Parity Difference: {statistical_parity_difference:.4f}") #output -> Statistical Parity Difference: 0.0039
```

```
Statistical Parity Difference: 0.0039
```

05

ML Test Pyramid

Testing Strategies

Unit Testing

Testing individual components (data pre-processing, feature extraction et al.)

Integration Testing

Verifying the interactions and compatibility b/w various components (data pipeline)

Model Testing

Examine the performance and the behavior of ML model (evaluating metrics like PRF scores)

Robustness Testing

Examine the system's ability to handle edge cases, outliers and adversarial inputs.

Deployment Testing

Validating ML system in production environment under real-world conditions

Ethical & Fairness Testing

Evaluation for bias, discrimination & adherence to ethical regulations

```python
@pytest.mark.parametrize("text, expected_sentiment", [
    ("The absolutely love Prague, it's so beautiful!", 1),
    ("The movie was awful.", 0),
    ("The movie was great, but the plot of the movie was not good.", pytest.param(None, marks=pytest.mark.xfail)),
    ("", None)
])
def test_sentiment_classification(sentiment_classifier, training_data, test_data, text, expected_sentiment):
    train_classifier(sentiment_classifier, *training_data)
    sentiment = sentiment_classifier.predict_sentiment(text)
    assert sentiment == expected_sentiment

def test_non_english_text(sentiment_classifier, training_data):
    train_classifier(sentiment_classifier, *training_data)
    text = "Je déteste ce film."
    sentiment = sentiment_classifier.predict_sentiment(text)
    assert sentiment in [0, 1]
```

06

Let’s plug in Explainability

Explainable Data

What data was used to train the model?

Explainable Predictions

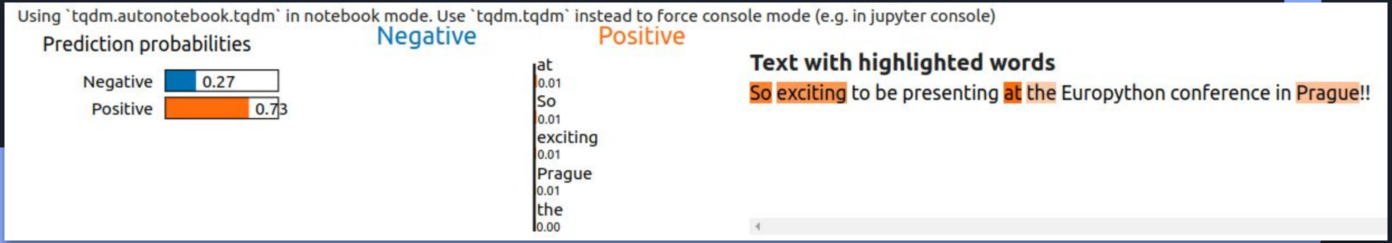
What features and weights were used for this particular task / prediction?

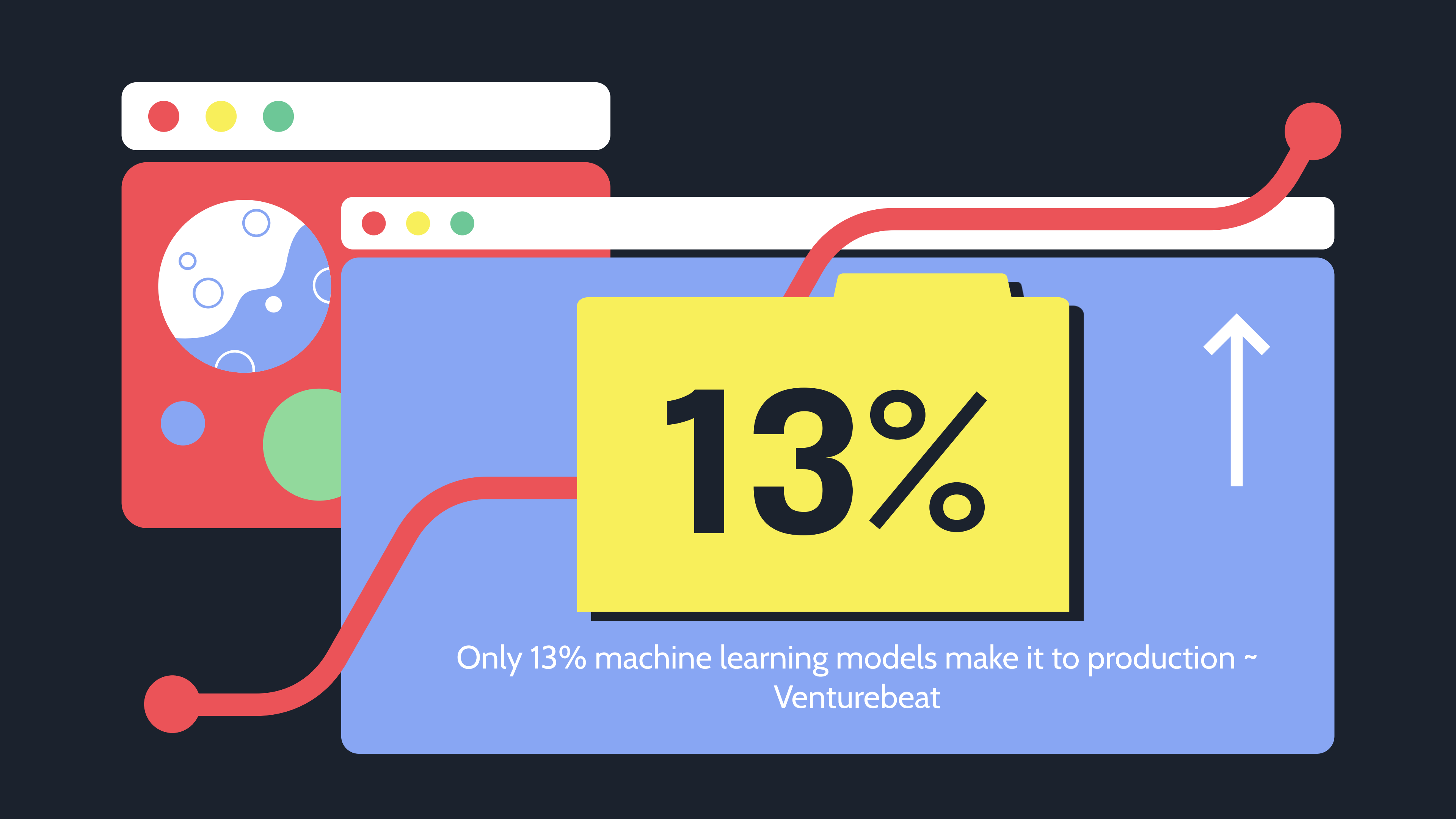
Explainable Algorithms

What are the individual layers and the thresholds for predictions?

```
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from lime.lime_text import LimeTextExplainer

# Sample training data
texts = [ "I love the beaches at Miami!!", "This movie is terrible", "The sounds so exciting!", "The plot was not good", "We have so many attendees, be it developers, CEO's, datascientists, researchers at the Europython conference!",
      "I love the night-life at Prague!"]
labels = [1, 0, 1, 0, 1, 1]

# Vectorize the data
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(texts)
```

```python
# Train a logistic regression classifier
model = make_pipeline(vectorizer, LogisticRegression())
model.fit(texts, labels)

# Text to explain
text_to_explain = "So exciting to be presenting at the Europython conference in Prague!!"

# LIME explanation
explainer = LimeTextExplainer(class_names=["Negative", "Positive"])
exp = explainer.explain_instance(text_to_explain, model.predict_proba, num_features=5)
exp.show_in_notebook(text=True)
```




13%
Only 13% machine learning models make it to production ~ Venturebeat

07

Technical Debts in ML Sys

DATA DEBT	This includes issues like low data quality, small datasets, and lack of representativeness.
ARCHITECTURE DEBT	An overly complex ML architecture which makes system hard to navigate, extend and debug.
ALGORITHMIC DEBT	Reliance on the outdated or suboptimal modelling assumptions / techniques.
TESTING DEBT	Lack of tests written to verify correct functionality, catch bugs, and refactors of the ML system.

08

Addressing technical debts

Review the ML pipeline

Check data quality issues

Examine model performance

Evaluate model explainability

Adopt automated testing practises

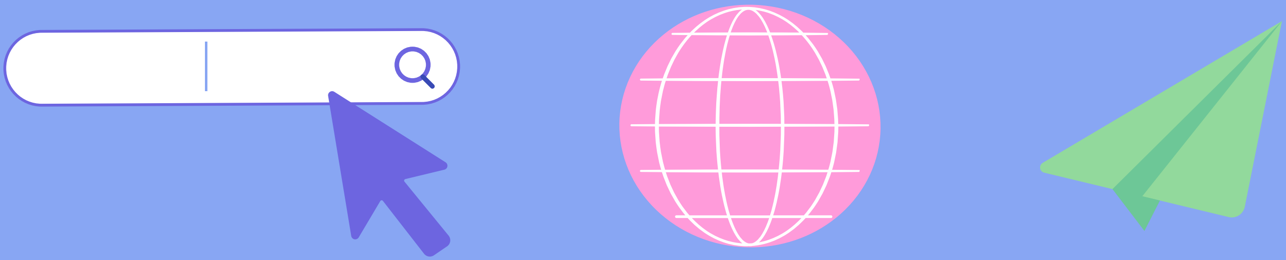
Perform regular code refactoring

09 Conclusion

- Building fair, accountable, and trustworthy machine learning systems is critical for ensuring that the benefits of these systems are widely accessible and enjoyed by all.
- By ensuring that our models are transparent and interpretable, we can detect and mitigate biases and unfairness before they cause harm.
- Monitoring and evaluation of these models is vital to maintain accountability and ensure that they continue to function as intended.

Alternative resources

- Book: https://fairmlbook.org/
- Course: Introduction to Deep Learning http://introtodeeplearning.com/
- Article: https://hdsr.mitpress.mit.edu/pub/f9kuryi8/release/8



“Garbage in, garbage out, so let’s sift through biases and purge them out! In the realm of fairness, one size never fits, it’s a kaleidoscope of perspectives and intricate bits.”

– Yours truly

@iamrashminagpal

Thank you!

@iamrashminagpal