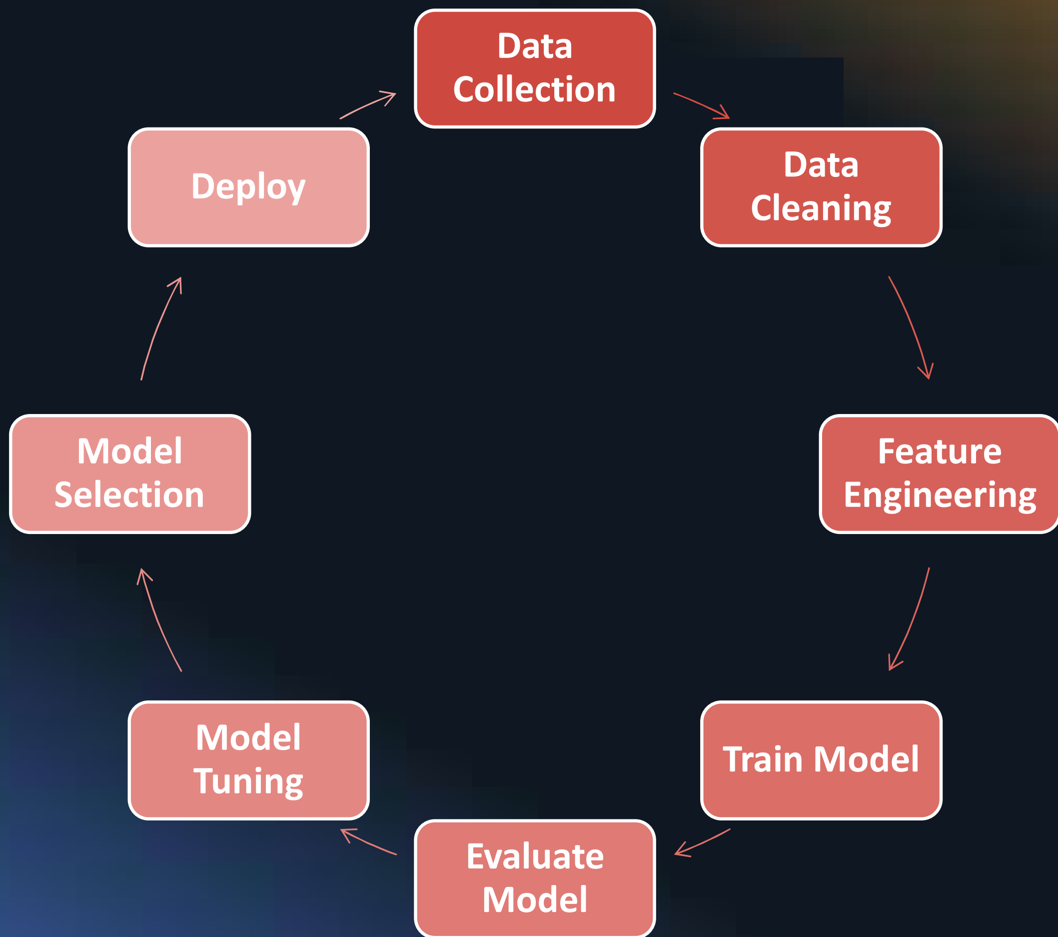
BUILD AN EXPLAINABLE AI MODEL WITH AWS SAGEMAKER STUDIO

SARBANI MAITI

VP DATA SCIENCE AT VEERSA TECHNOLOGY

EMAIL: SARBANIIIITB2020@GMAIL.COM

Machine Learning Lifecycle



MACHINE LEARNING LIFECYCLE IN PRODUCTION

Once the model is deployed, we need a production ready solution to serve the consumer applications.

Availability

- Availability of the ML system is critical for production system
- Onboarding the right kind of tool and technologies which are easy to adopt as per the change management system of the organization.
- Maintenance and reusability of the tools and models.
- Tracking, monitoring, alerting and feedback loop are other important aspects of the model in production

Scalability

- Model must support automatic scaling in production. Autoscaling dynamically adjusts the number of instances provisioned for a model in response to changes in your workload.
- When the workload increases, autoscaling brings more instances online.
- When the workload decreases, autoscaling removes unnecessary instances so that users don't pay for provisioned instances that are not in use.

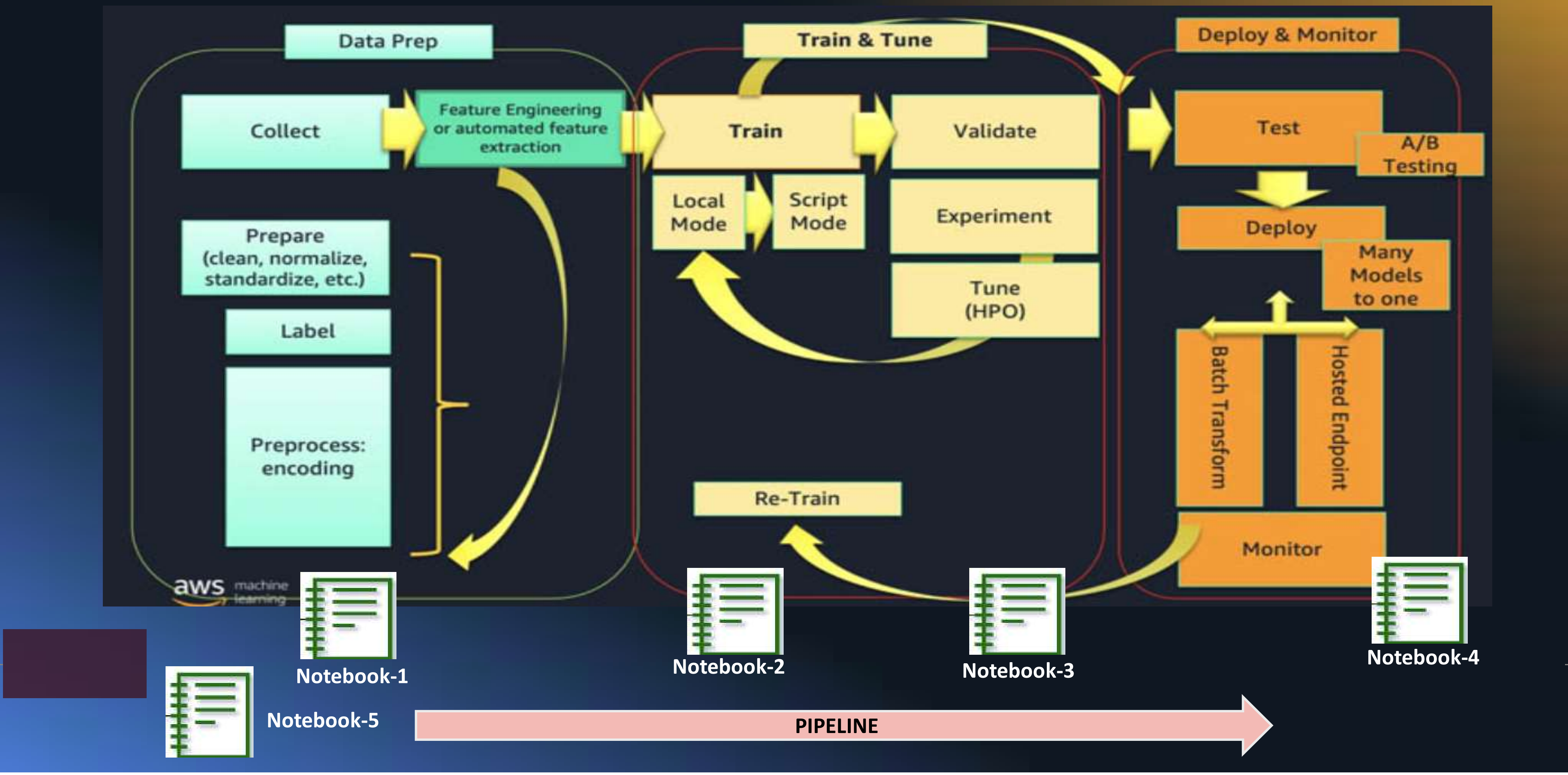
Security

- The model has to be compliant as per the regulatory requirement.
- Create environments with the least privileged access to sensitive data
- Protect & Encrypt sensitive data
- Audit and trace activity in your environment
- Reproduce results in your environment by tracking the lineage of ML artifacts throughout the lifecycle and using source and version control tools

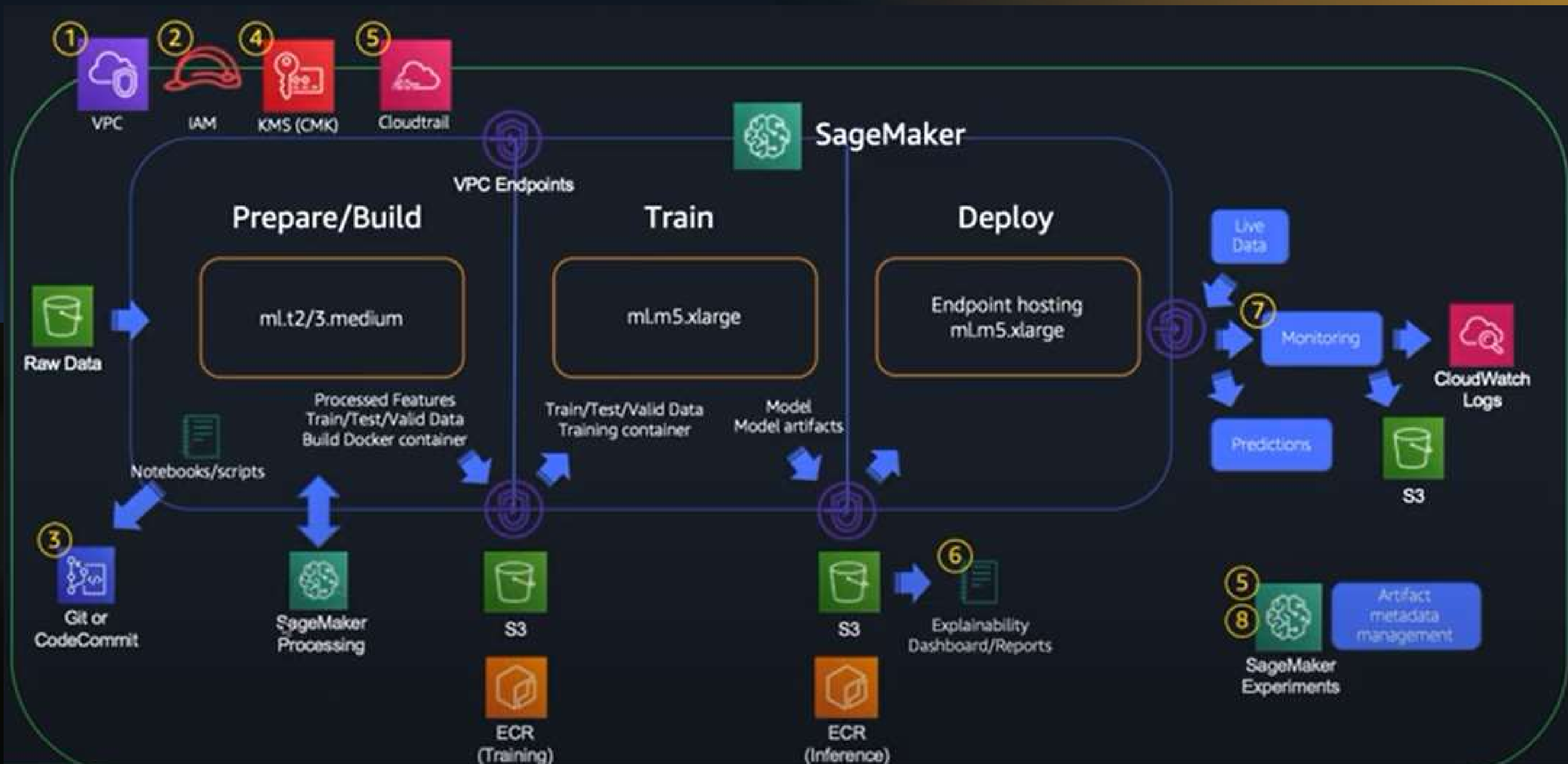
SECURE & COMPLAINT ML WORKFLOW

1	COMPUTE & NETWORK ISOLATION	Deploy SageMaker in a VPC with no Internet access
2	AUTHENTICATION & AUTHORIZATION	Provide single user access to Jupyter over IAM
3	ARTIFACT MANAGEMENT	Enable private Git integration, lifecycle config, and versioning
4	DATA ENCRYPTION	Encrypt data at motion and at rest across all ML workflow
5	TRACEABILITY & AUDITABILITY	Trace model lineage, and audit all API calls and data events
6	EXPLAINABILITY & INTERPRETABILITY	Explain predictions with feature importance and SHAP values
7	REAL-TIME MODEL MONITORING	Monitor the performance of a productionized model
8	REPRODUCIBILITY	Reproduce the model and results based on saved artifacts

REFERENCE ML SYSTEM



SAGEMAKER SOLUTION



Amazon SageMaker overview

Amazon SageMaker

PREPARE

SageMaker Ground Truth
Label training data for machine learning

SageMaker Data Wrangler NEW
Aggregate and prepare data for machine learning

SageMaker Processing
Built-in Python, BYO R/Spark

SageMaker Feature Store
Store, update, retrieve, and share features

SageMaker Clarify
Detect bias and understand model predictions

BUILD

SageMaker Studio Notebooks
Jupyter notebooks with elastic compute and sharing

Built-in and Bring your-own Algorithms
Dozens of optimized algorithms or bring your own

Local Mode
Test and prototype on your local machine

SageMaker Autopilot
Automatically create machine learning models with full visibility

SageMaker JumpStart
Pre-built solutions for common use cases

TRAIN & TUNE

Managed Training
Distributed infrastructure management

SageMaker Experiments
Capture, organize, and compare every step

Automatic Model Tuning
Hyperparameter optimization

Distributed Training Libraries NEW
Training for large datasets and models

SageMaker Debugger NEW
Debug and profile training runs

Managed Spot Training
Reduce training cost by 90%

DEPLOY & MANAGE

Managed Deployment
Fully managed, ultra low latency, high throughput

Kubernetes & Kubeflow Integration
Simplify Kubernetes-based machine learning

Multi-Model Endpoints
Reduce cost by hosting multiple models per instance

SageMaker Model Monitor
Maintain accuracy of deployed models

SageMaker Edge Manager NEW
Manage and monitor models on edge devices

SageMaker Pipelines NEW
Workflow orchestration and automation

SageMaker Studio
Integrated development environment (IDE) for ML

Build ML models : full to no customization

CUSTOM BUILD MODELS

- Your own machine learning code
- Opensource/ proprietary implementations or frameworks

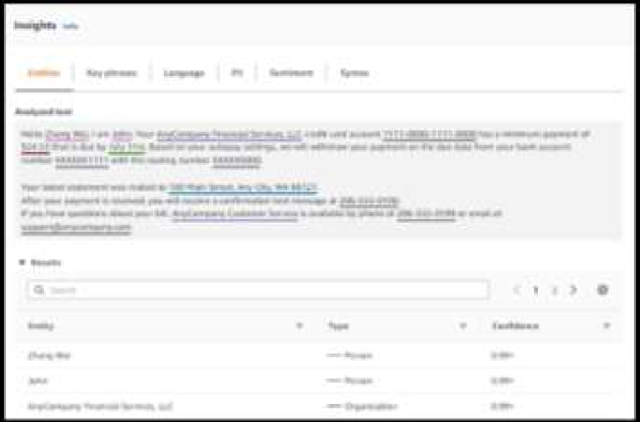
- More time to build
- More costly
- High level of ML expertise
- Most flexible

Build ML models : full to no customization

PRE-TRAINED MODELS: WHY TRAIN WHEN YOU CAN FINE TUNE?

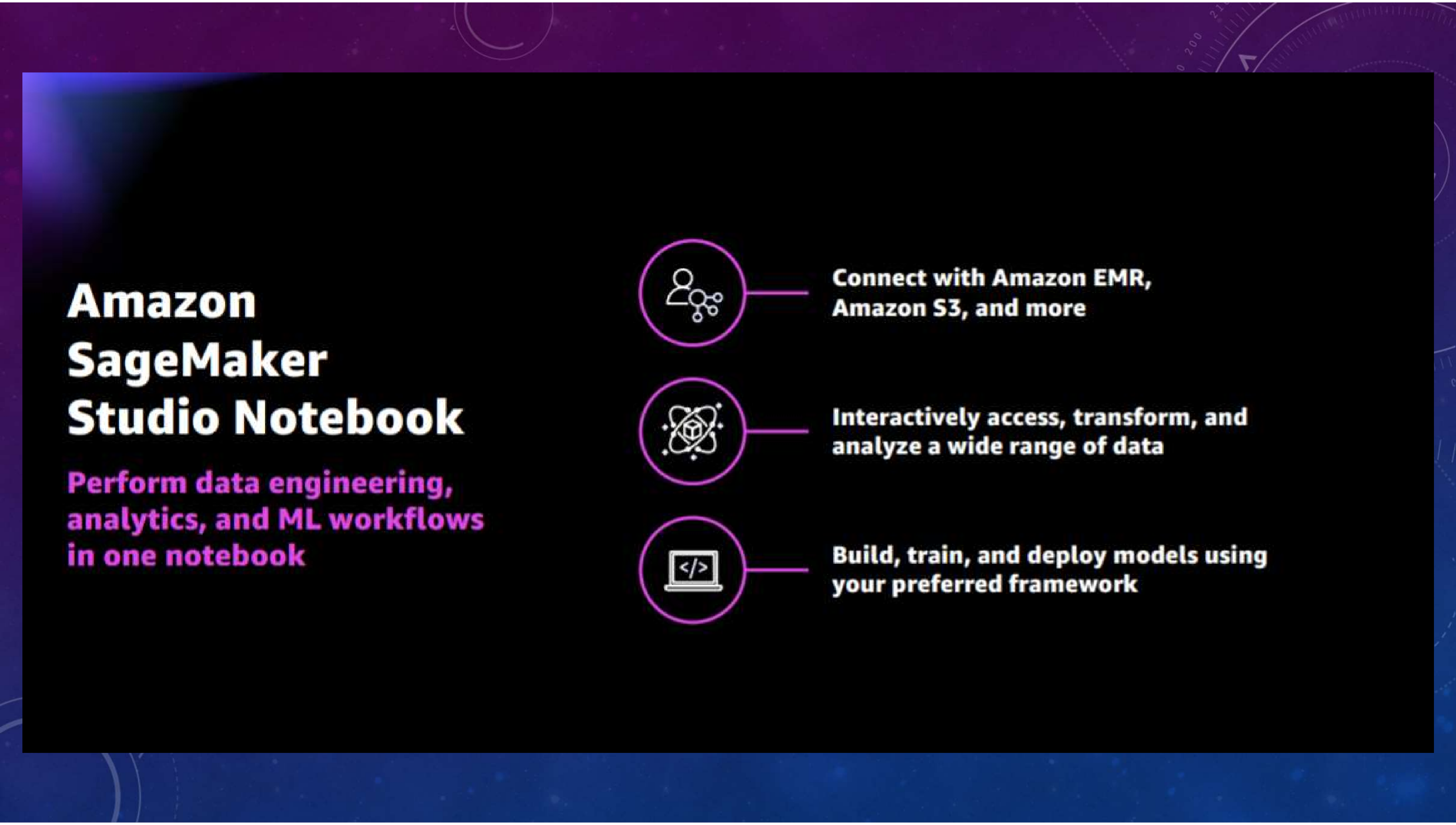
- Pre-trained model: a model that was trained on a large benchmark dataset to solve a problem similar to the one you want to solve.
  - Use the deployed version via an API call (managed)
  - Deploy directly (self-managed)
  - Fine tune (Managed or self managed)

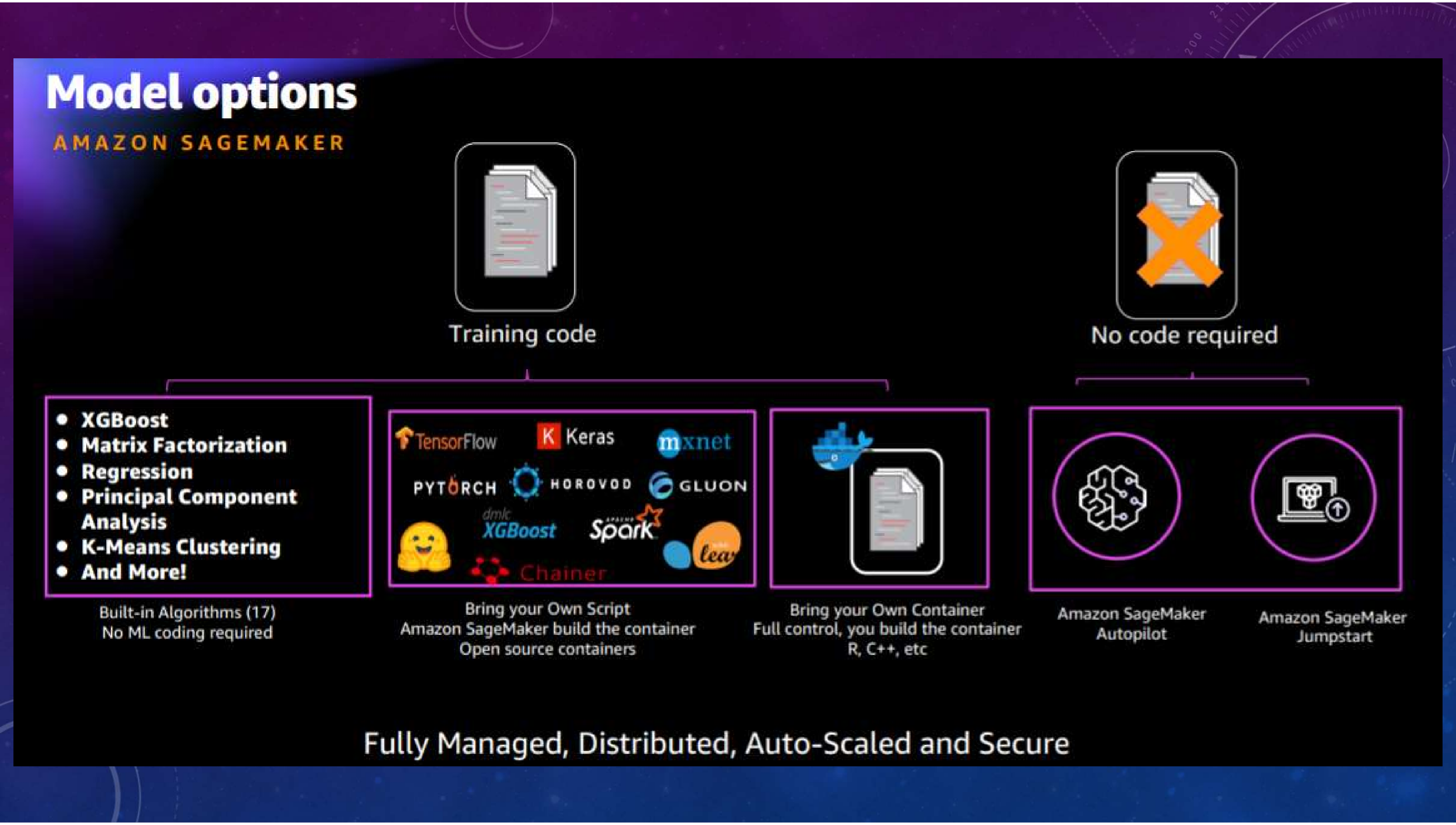
Amazon Comprehend real time text analysis



Amazon Rekognition label detection

- Saves time
- Saves experimentation and cost
- ML expertise
- None to some flexibility


Amazon SageMaker Studio Notebook
Perform data engineering, analytics, and ML workflows in one notebook
Connect with Amazon EMR, Amazon S3, and more
Interactively access, transform, and analyze a wide range of data
Build, train, and deploy models using your preferred framework


Model options
AMAZON SAGEMAKER
Training code
No code required
XGBoost
Matrix Factorization
Regression
Principal Component Analysis
K-Means Clustering
And More!
TensorFlow
Keras
mxnet
PYTORCH
HOROVOD
GLUON
dmlc XGBoost
Spark
Chainer
Built-in Algorithms (17)
No ML coding required
Bring your Own Script
Amazon SageMaker build the container
Open source containers
Bring your Own Container
Full control, you build the container
R, C++, etc
Amazon SageMaker Autopilot
Amazon SageMaker Jumpstart
Fully Managed, Distributed, Auto-Scaled and Secure

Amazon SageMaker in-built algorithms

Amazon SageMaker
has built-in algorithms
or bring your own

Computer vision
Image classification | Object detection | Semantic segmentation

Topic modeling
LDA | NTM

Classification
Linear Learner | XGBoost | KNN

Recommendation
Factorization machines

Forecasting
DeepAR

Working with text
BlazingText | Supervised | Unsupervised

Regression
Linear Learner | XGBoost | KNN

Clustering
KMeans

Sequence translation
Seq2Seq

Anomaly detection
Random cut forests | IP Insights

Feature reduction
PCA

Amazon SageMaker in-built algorithms

SAMPLE CODE

```
region=boto3.Session().region_name
container = sagemaker.image_uris.retrieve(region=region, framework='xgboost', version='latest')
print( "Using SageMaker XGBoost container: {}, ({})".format (container, region))
```

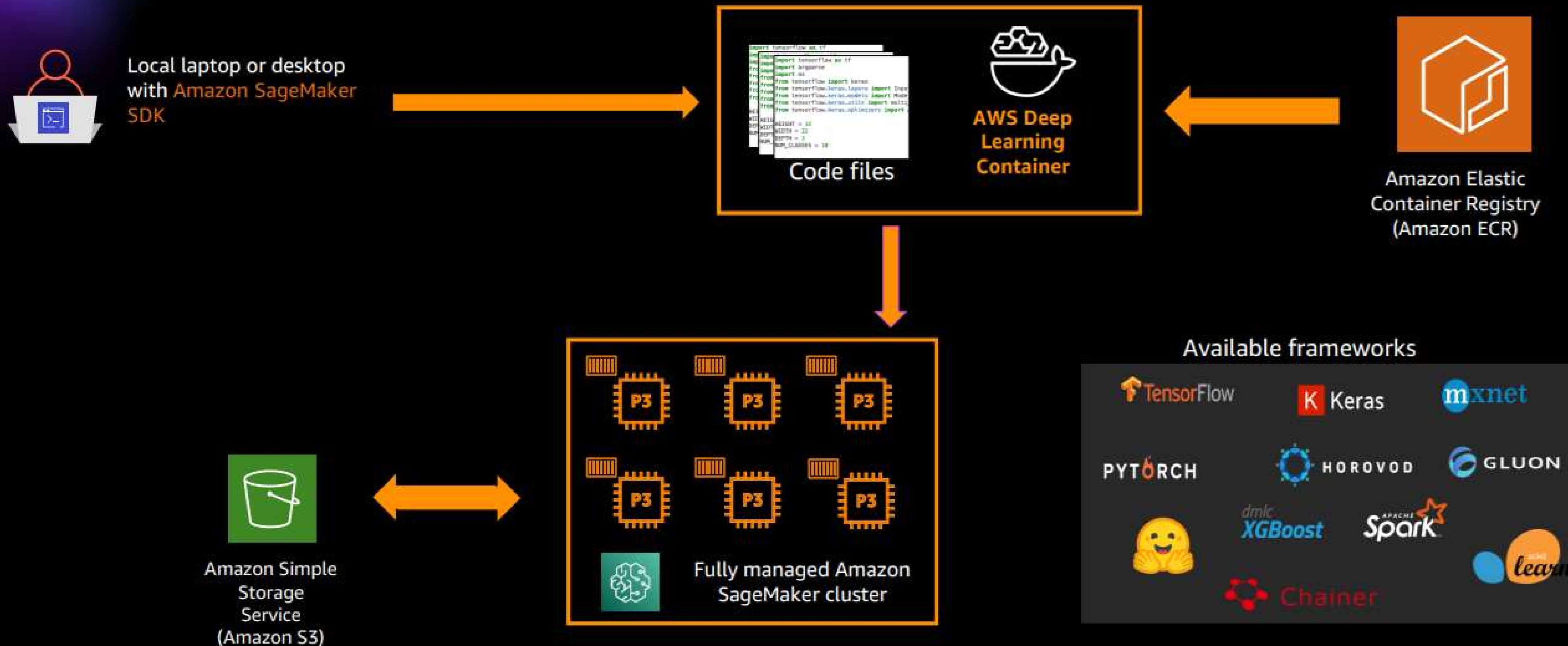
```
using SageMaker XGBoost container: 544295431143.dkr.ecr.ap-southeast-2.amazonaws.com/xgboost:latest, (ap-southeast-2)
```

```
sess = sagemaker.Session()

xgb = sagemaker.estimator.Estimator(container,
                                    role,
                                    instance_count=1,
                                    instance_type='ml.m4.xlarge',
                                    output_path='s3://{}/{}/output'.format(bucket, prefix),
                                    sagemaker_session=sess)
xgb.set_hyperparameters(max_depth=5,
                        eta=0.2,
                        gamma=4,
                        min_child_weight=6,
                        subsample=0.8,
                        silent=0,
                        objective='binary:logistic',
                        num_round=100)

xgb.fit({'train': s3_input_train, 'validation': s3_input_validation})
```

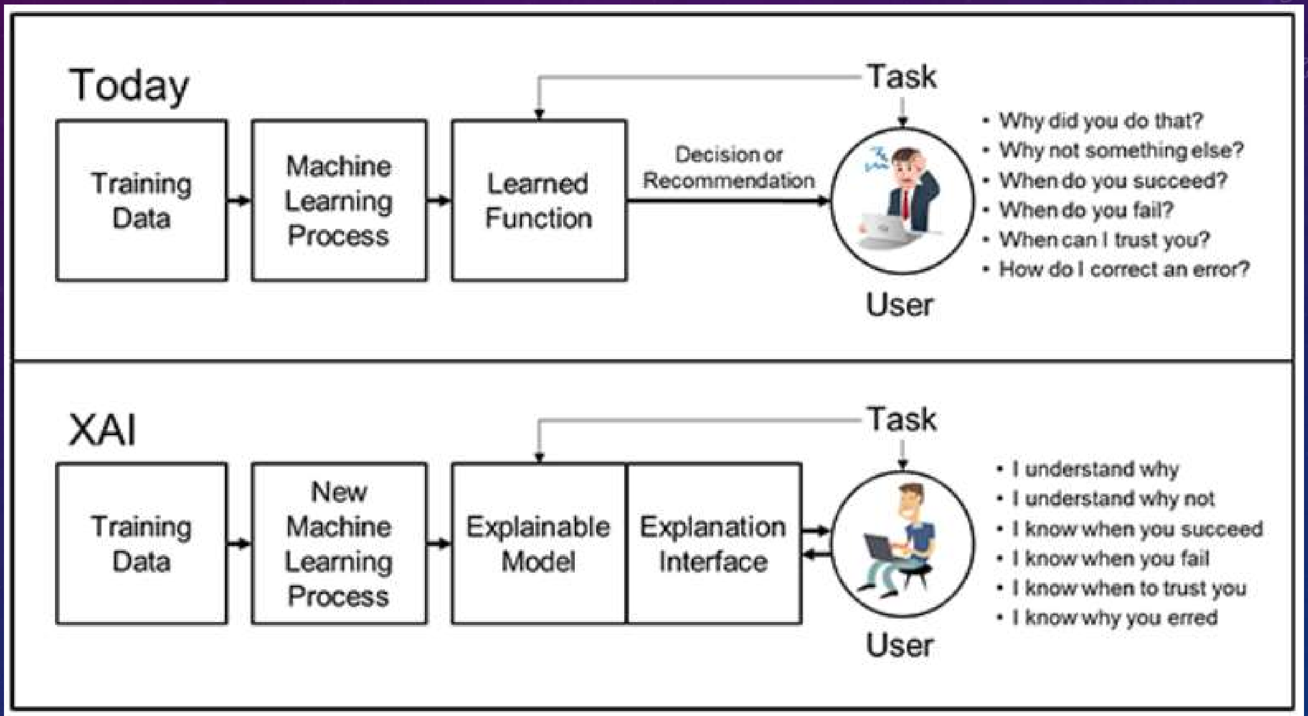
Amazon Elastic Container Registry (Amazon ECR)

Bring your own script ('script mode')

HIGH LEVEL WORKFLOW



Understand How Machines Make Decision



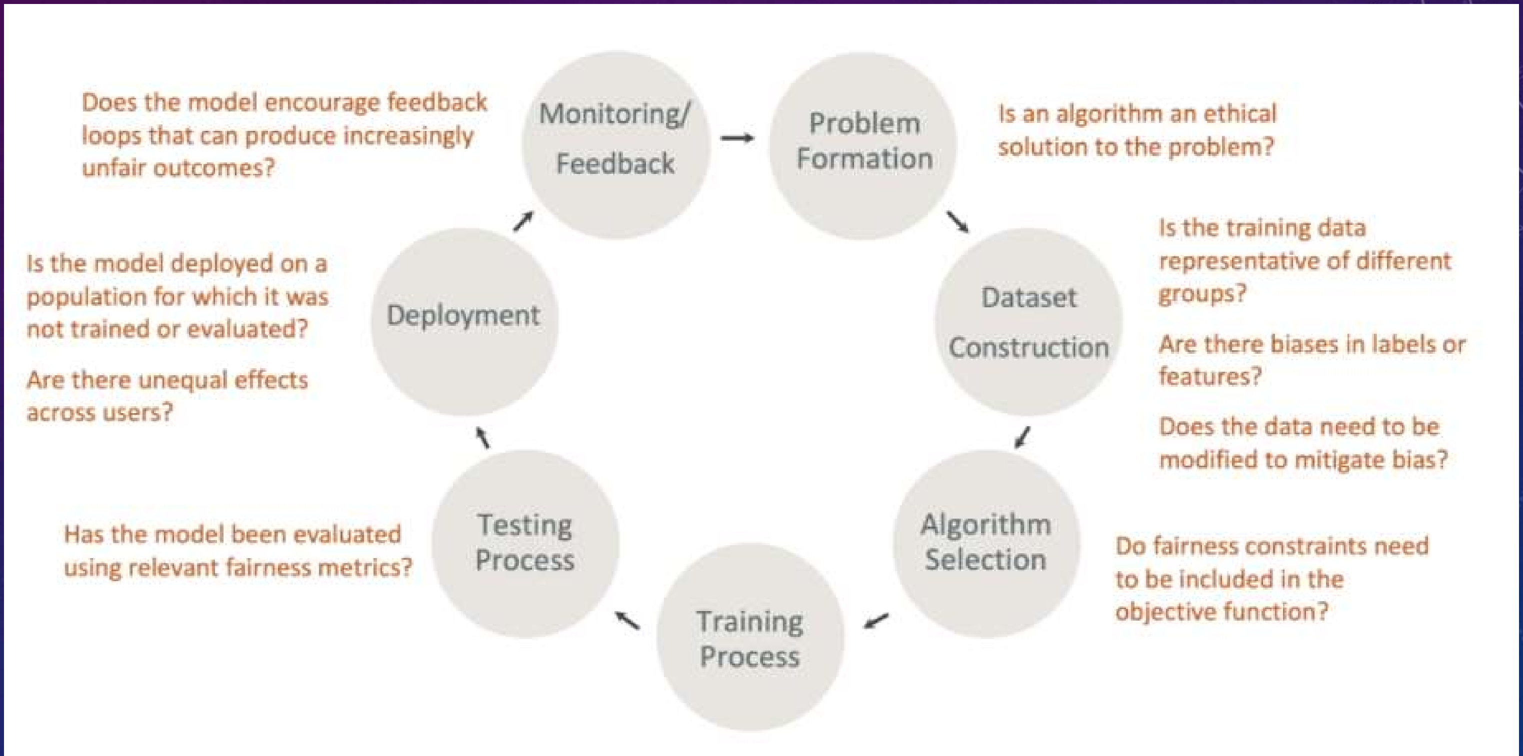
The diagram Source: https://www.darpa.mil/program/explainable-artificial-intelligence

Understand How Machines Make Decision

AI-based systems are disrupting almost every industry – healthcare, finances, education, retail, media, marketing, social, economy, HR system, etc, etc. and helping us to make crucial decisions that are impacting millions of lives.

Hence it is important to understand how these decisions are made by the AI system.

Machine Learning Lifecycle



https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-fairness-and-explainability.html

Explain the Model Decision

Simple Algorithms

- Bayesian classifiers,
- Decision trees,
- Linear Regression,
- Logistic Regression etc ..

➢ possess a certain degree of interpretability, traceability, visibility, and transparency in their decision-making process at the cost of the performance

Complex Algorithms

- PCA,
- Deep Neural Network,
- Ensemble methods,
- Random forests etc

➢ sacrifice their explainability to achieve high performance and accuracy

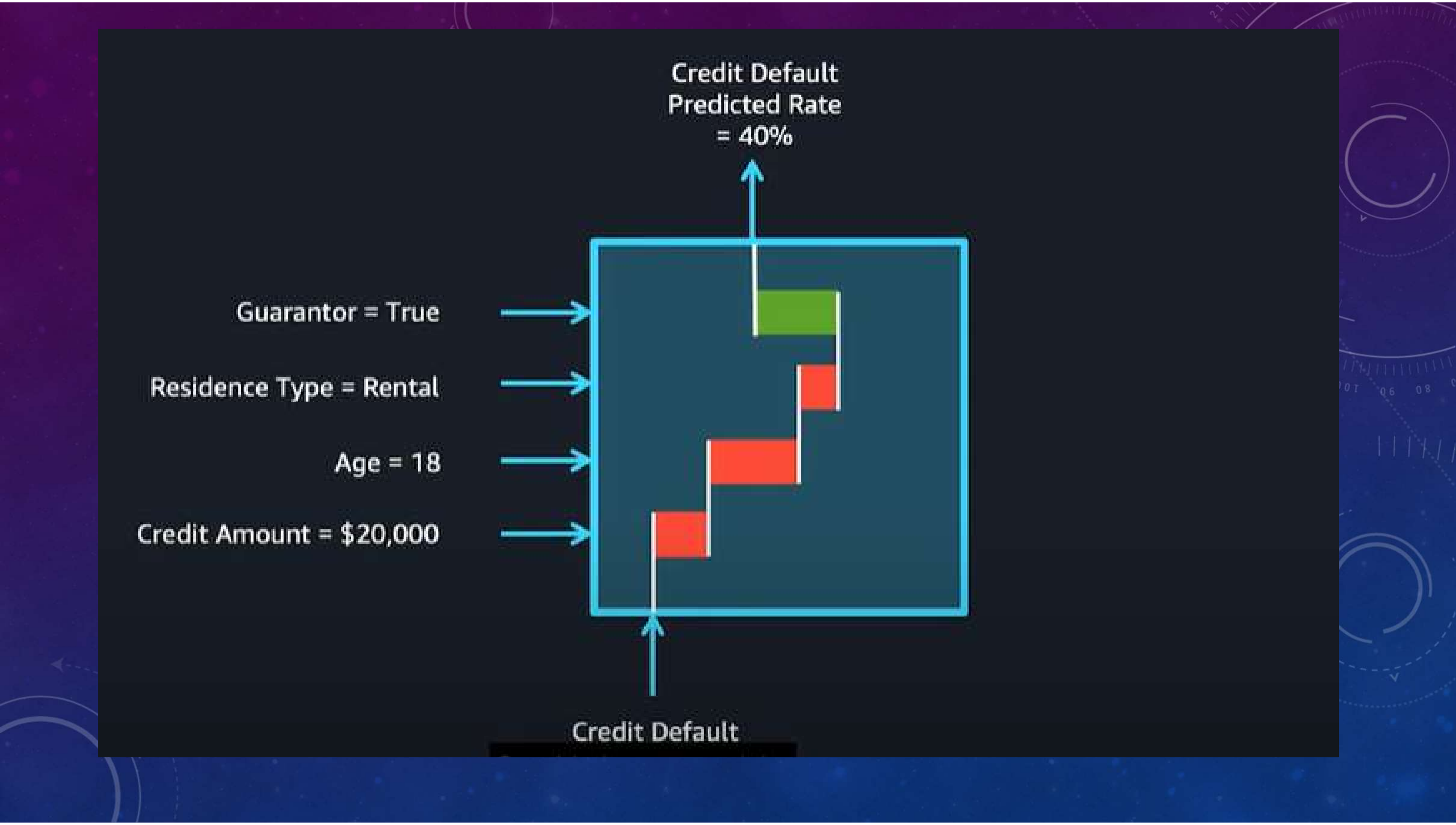
AWS Sagemaker clarify

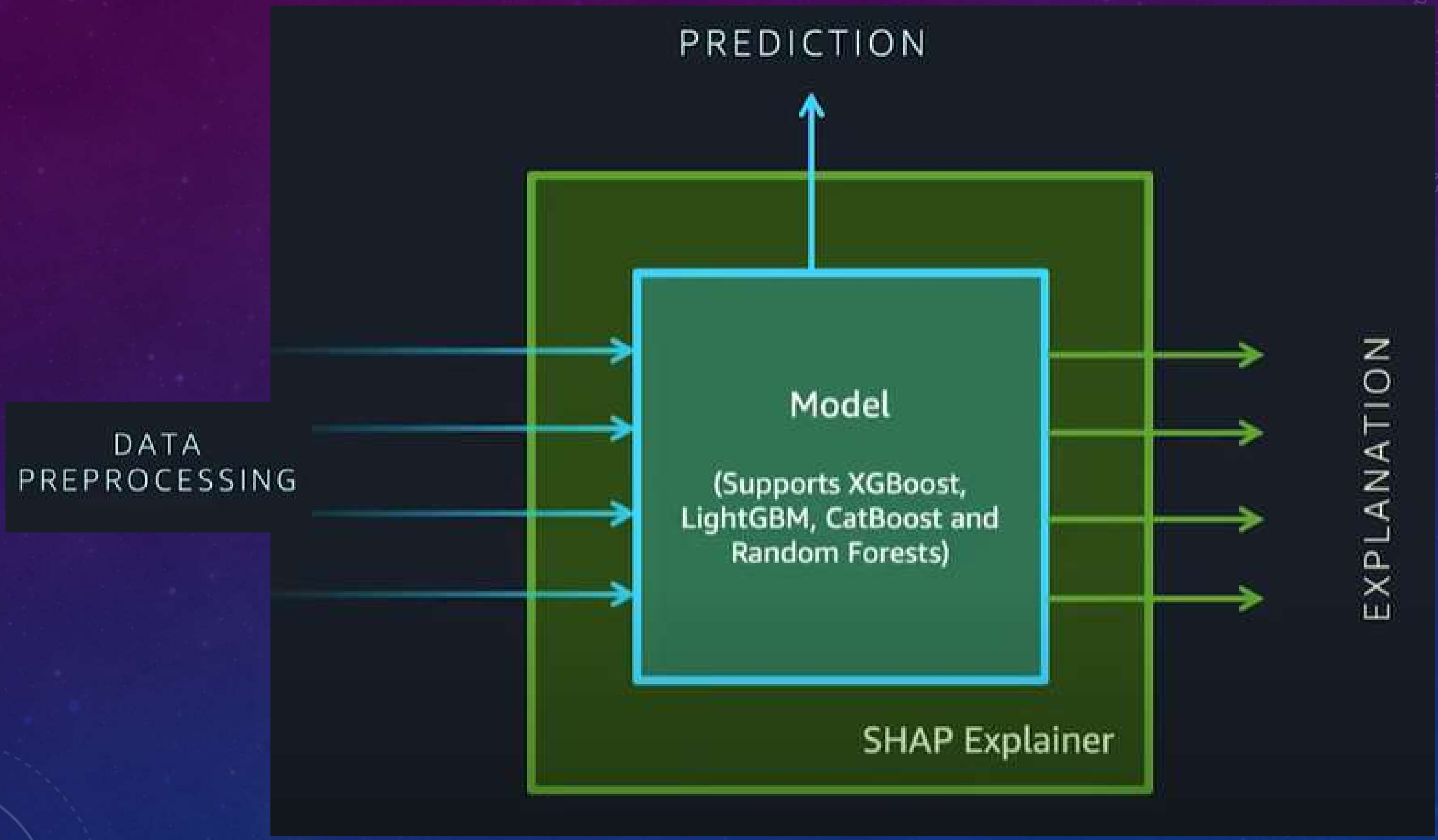
- Detecting dataset bias

  ✓ Pre Training & Post Training Bias Detection
  ✓ Measure bias using a variety of statistical metrics.
  ✓ Help in early detection of bias in dataset

- Detecting model bias

  ✓ Explain how feature values contribute to the predicted outcome, both for the model overall and for individual predictions.
  ✓ Run a SageMaker Clarify analysis, which includes automatic deployment to a temporary endpoint, and computation of bias metrics using our model and dataset.
  ✓ By computing these metrics, we can figure out if trained model has similar predictive behavior across groups.


Credit Default
Predicted Rate
= 40%
Guarantor = True
Residence Type = Rental
Age = 18
Credit Amount = $20,000
Credit Default


PREDICTION
DATA
PREPROCESSING
Model
(Supports XGBoost,
LightGBM, CatBoost and
Random Forests)
SHAP Explainer
EXPLANATION

SHAP stands for SHapley Additive exPlanations. 'Shapley' relates to a game theoretic concept called Shapley values that is used to create the explanations.

A Shapley value describes the marginal contribution of each 'player' when considering all possible 'coalitions'.

Using this in a machine learning context, a Shapley value describes the marginal contribution of each feature when considering all possible sets of features.

'Additive' relates to the fact that these Shapley values can be summed together to give the final model prediction.

As an example, we might start off with a baseline credit default risk of 10%. Given a set of features, we can calculate the Shapley value for each feature. Summing together all the Shapley values, we might obtain a cumulative value of +30%. Given the same set of features, we therefore expect our model to return a credit default risk of 40% (i.e. 10% + 30%).

Shapeley Values

WITH GUARANTOR		WITHOUT GUARANTOR	
Guarantor = True	✓	Guarantor = ~~True~~ = False	✗
Residence Type = Rental	✓	Residence Type = Rental	✓
Age = 18	✓	Age = 18	✓
Credit Amount = $20,000	✓	Credit Amount = $20,000	✓
Credit Default Risk = 40%		Credit Default Risk = 60%	
		Effect of Guarantor = **-20%**	

WITH GUARANTOR		WITHOUT GUARANTOR	
Guarantor = True	✓	Guarantor = ~~True~~ = False	✗
Residence Type = ~~Rental~~ = Own	✗	Residence Type = ~~Rental~~ = Own	✗
Age = 18	✓	Age = 18	✓
Credit Amount = $20,000	✓	Credit Amount = $20,000	✓
Credit Default Risk = 10%		Credit Default Risk = 20%	
		Effect of Guarantor = **-10%**, -20%	

WITH GUARANTOR		WITHOUT GUARANTOR	
Guarantor = True	✓	Guarantor = ~~True~~ = False	✗
Residence Type = ~~Rental~~ = Own	✗	Residence Type = ~~Rental~~ = Own	✗
Age = ~~18~~ = 45	✗	Age = ~~18~~ = 45	✗
redit Amount = ~~$20,000~~ = $5,000	✗	Credit Amount = ~~$20,000~~ = $5,000	✗
Credit Default Risk = 5%		Credit Default Risk = 45%	

Effect of Guarantor = -10%, -15%, -25%, -5%. -15%, -15%, **-10%, -20%**

Overall Effect of Guarantor = SHAP Value for Guarantor = -17%

Shapeley Values

WITH GUARANTOR

Guarantor = True ✓
Residence Type = ~~Rental~~ = Own ✗
Age = 18 ✓
Credit Amount = ~~$20,000~~ = $5,000 ✗
Credit Default Risk = 20%

WITHOUT GUARANTOR

Guarantor = ~~True~~ = False ✗
Residence Type = ~~Rental~~ = Own ✗
Age = 18 ✓
Credit Amount = ~~$20,000~~ = $5,000 ✗
Credit Default Risk = 45%
Effect of Guarantor = -25%, -5%, -15%, -15%, -10%, -20%

WITH GUARANTOR

Guarantor = True ✓
Residence Type = ~~Rental~~ = Own ✗
Age = ~~18~~ = 45 ✗
redit Amount = ~~$20,000~~ = $5,000 ✗
Credit Default Risk = 5%

WITHOUT GUARANTOR

Guarantor = ~~True~~ = False ✗
Residence Type = ~~Rental~~ = Own ✗
Age = ~~18~~ = 45 ✗
Credit Amount = ~~$20,000~~ = $5,000 ✗
Credit Default Risk = 45%
Effect of Guarantor = -10%, -15%, -25%, -5%. -15%, -15%, -10%, -20%

Overall Effect of Guarantor = SHAP Value for Guarantor = -17%

Explain Credit Decision with Sagemaker Clarify

- Classify credit applications and predict whether the credit would be payed back or not

- Bank to Reduce the risk of losing money due to unpaid credits

- Also reduce the risk of denying trustworthy customers credit which has a set of negative impacts.

AWS Services

As part of the solution, the following services are used:

AWS Lambda: Used to generate a synthetic credits dataset and upload to Amazon S3.

AWS Glue: Used to crawl datasets, and transform the credits dataset using Apache Spark.

Amazon S3: Used to store datasets and the outputs of the AWS Glue Job.

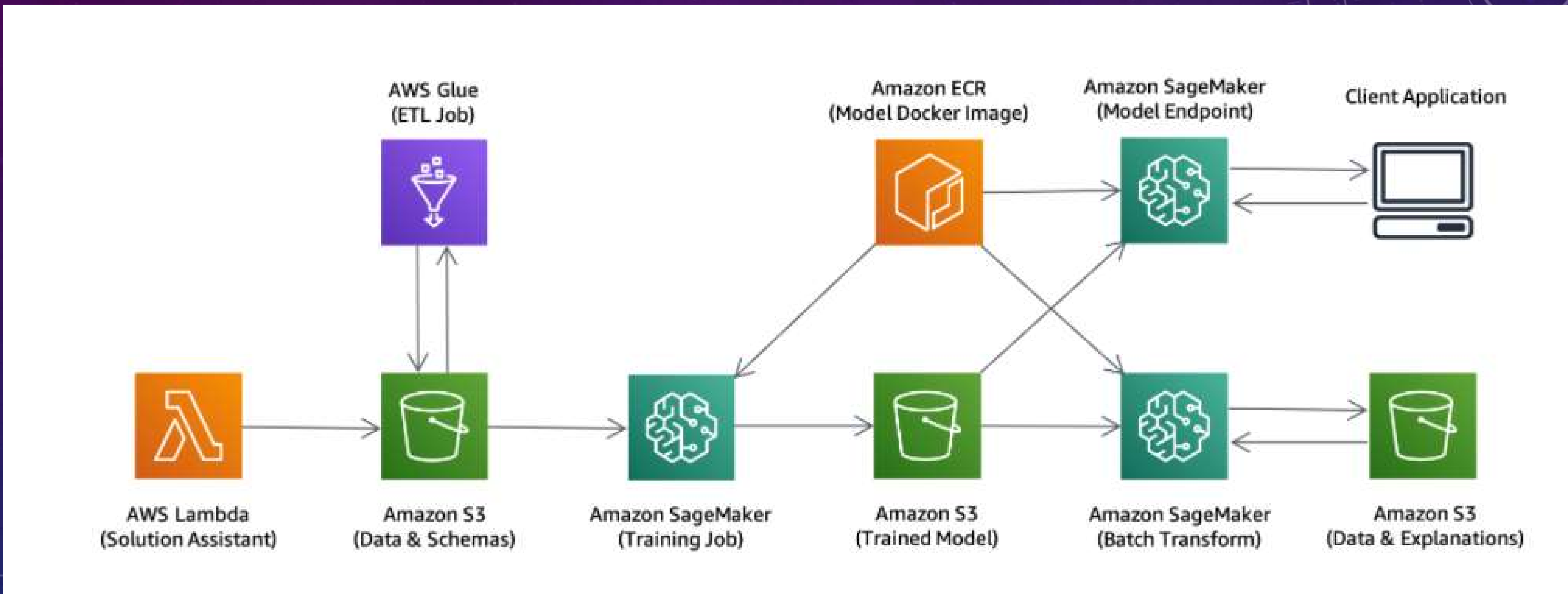
Amazon SageMaker Notebook: Used to train the LightGBM model.

Amazon ECR: Used to store the custom Scikit-learn + LightGBM training environment.

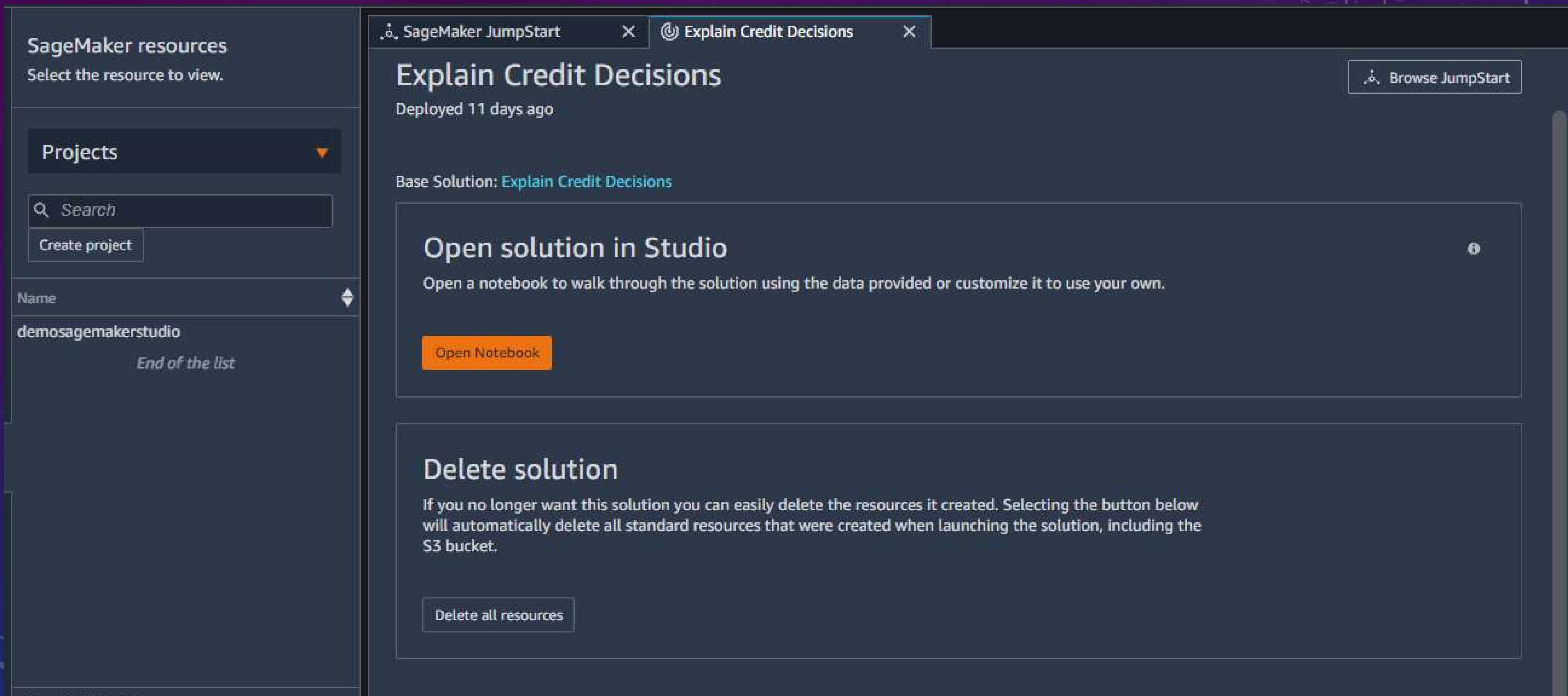
Amazon SageMaker Endpoint: Used to deploy the trained model and SHAP explainer.

Amazon SageMaker Batch Transform: Used to compute explanations in batch.

Architecture



AWS Sagemaker Studio



Stages

- Our solution is split into the following stages, and each stage has it's own notebook:
- Introduction: We take a high-level look at the solution components.
- Datasets: We prepare a dataset for machine learning using AWS Glue.
- Training: We train a LightGBM model using Amazon SageMaker, so we have an example trained model to explain.
- Endpoint: We deploy the model explainer to a HTTP endpoint using Amazon SageMaker and visualize the explanations.
- Batch Transform: We use Amazon SageMaker Batch Transform to obtain explanations for our complete dataset.
- Dashboard: We develop a dashboard for explanations using Amazon SageMaker and Streamlit.
- Conclusion: We wrap things up and discuss how to clean up the solution.

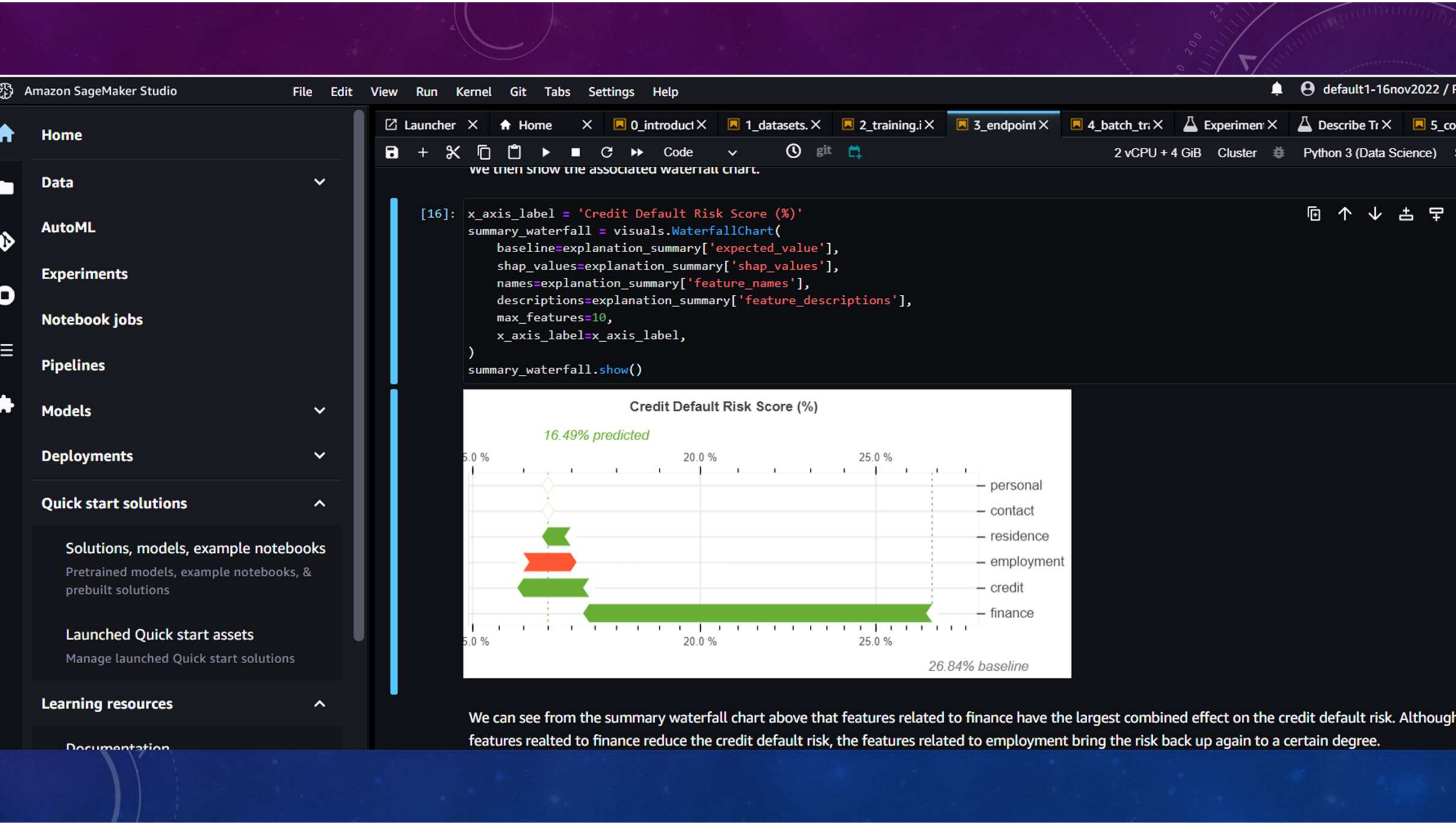
Model Deployment

- Once the SageMaker training job is completed, a number of trained model artifacts are stored in S3 bucket
- Retrive the model data (i.e. model.tar.gz) from the most recent trained model
- Define the model using SKLearnModel() to deploy which includes the explainer logic.
- Calling model.deploy() will start a container to host the model.
- Entities are used ->

```
entities = [
   'data',
   'features',
   'descriptions',
   'prediction',
   'explanation_shap_values',
   'explanation_shap_interaction_values'
]
```

Model Explanation

- Call explainer.predict with features (for a credit application) to obtain a prediction and explanation.
- Visualize Explanations with bokeh & waterfall chart.
- A waterfall chart can be used to show the cumulative effect of each feature.
- Green arrows indicate that the feature *decreased* the predicted credit default risk for the individual credit application.
- While red arrows indicate that the feature *increased* the predicted credit default risk for the individual credit application.
- After all features have been considered, we reach the final predicted credit default risk (at the top of the chart).

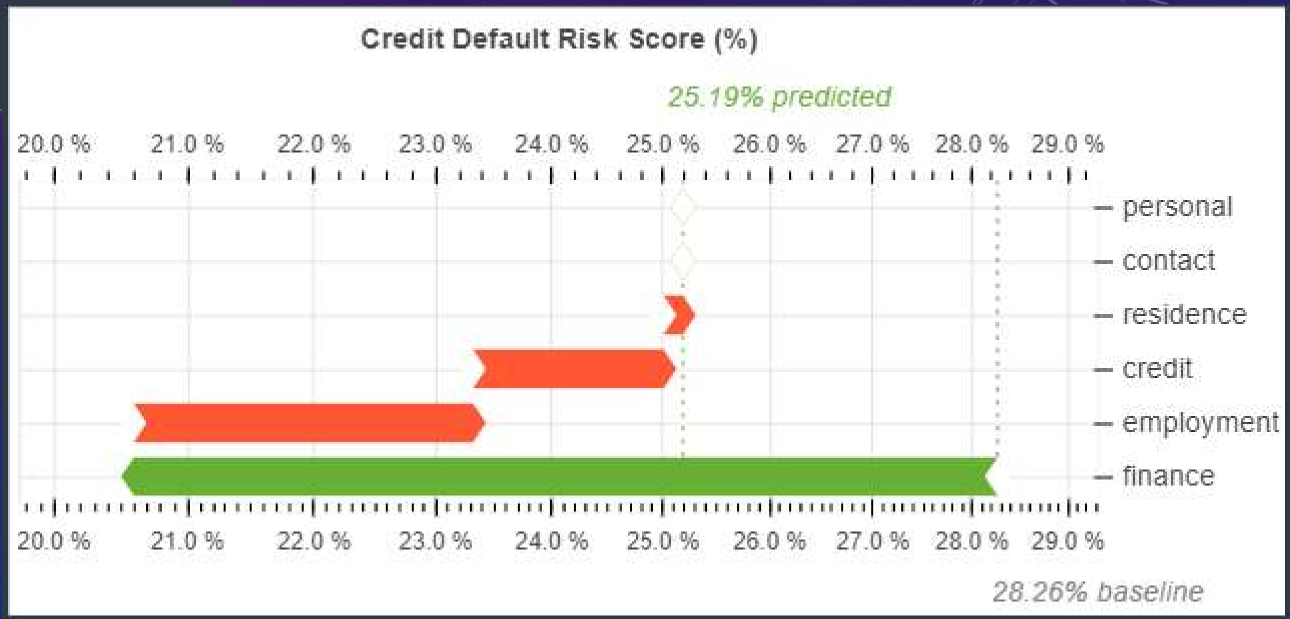

Amazon SageMaker Studio
File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help
default1-16nov2022 /
Home
Data
AutoML
Experiments
Notebook jobs
Pipelines
Models
Deployments
Quick start solutions
Solutions, models, example notebooks
Pretrained models, example notebooks, & prebuilt solutions
Launched Quick start assets
Manage launched Quick start solutions
Learning resources
Launcher
Home
0_introduct
1_datasets.
2_training.i
3_endpoint
4_batch_tr
Experimen
Describe Tr
Code
2 vCPU + 4 GiB
Cluster
Python 3 (Data Science)
[16]:
x_axis_label = 'Credit Default Risk Score (%)'
summary_waterfall = visuals.WaterfallChart(
    baseline=explanation_summary['expected_value'],
    shap_values=explanation_summary['shap_values'],
    names=explanation_summary['feature_names'],
    descriptions=explanation_summary['feature_descriptions'],
    max_features=10,
    x_axis_label=x_axis_label,
)
summary_waterfall.show()
Credit Default Risk Score (%)
16.49% predicted
20.0 %
25.0 %
personal
contact
residence
employment
credit
finance
26.84% baseline
We can see from the summary waterfall chart above that features related to finance have the largest combined effect on the credit default risk. Althoug
features realted to finance reduce the credit default risk, the features related to employment bring the risk back up again to a certain degree.

Model Explanation

```
x_axis_label = 'Credit Default Risk Score (%)'
summary_waterfall = visuals.WaterfallChart(
    baseline=explanation_summary['expected_value'],
    shap_values=explanation_summary['shap_values'],
    names=explanation_summary['feature_names'],
    descriptions=explanation_summary['feature_descriptions'],
    max_features=10,
    x_axis_label=x_axis_label,
)
summary_waterfall.show()
```

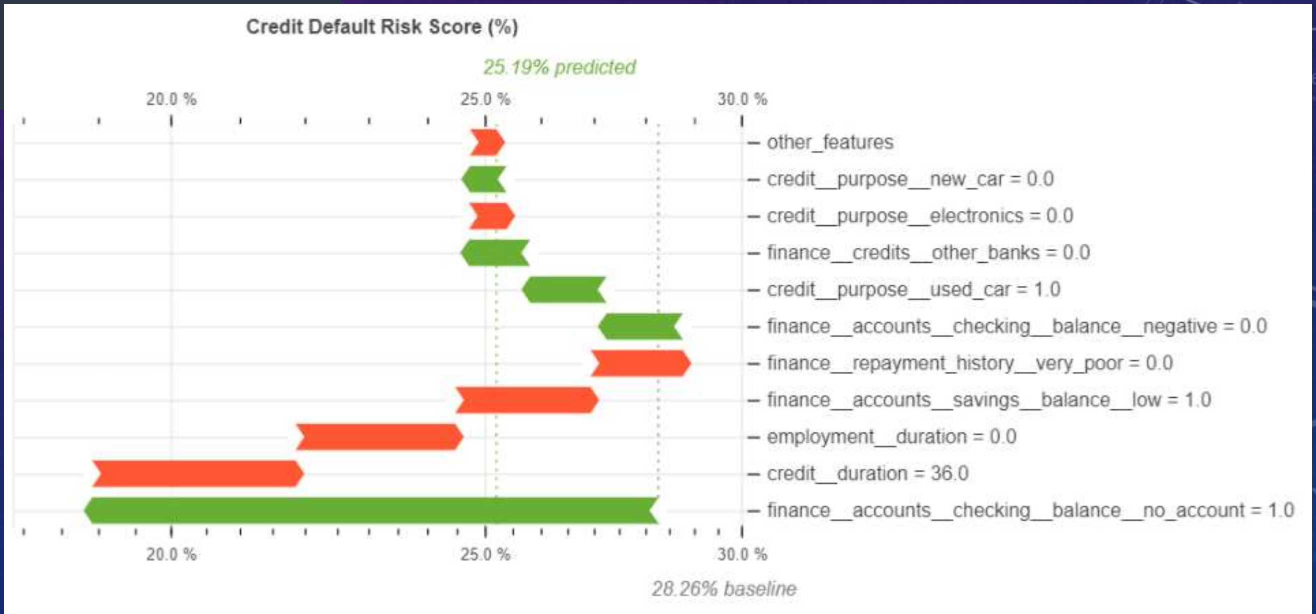
We can see from the summary waterfall chart that features related to finance have the largest combined effect on the credit default risk.

Although features related to finance reduce the credit default risk, the features related to employment bring the risk back up again to a certain degree.



Explain Prediction

```
detailed_waterfall = visuals.WaterfallChart(
    baseline=explanation['expected_value'],
    shap_values=explanation['shap_values'],
    names=explanation['feature_names'],
    feature_values=explanation['feature_values'],
    descriptions=explanation['feature_descriptions'],
    max_features=10,
    x_axis_label=x_axis_label
)
detailed_waterfall.show()
```



Explain Prediction

- Detailed waterfall chart shows that
  - Not having a checking account with the same bank indicates a lower credit default risk.
  - The credit to purchase a used car is associated with a lower credit default risk

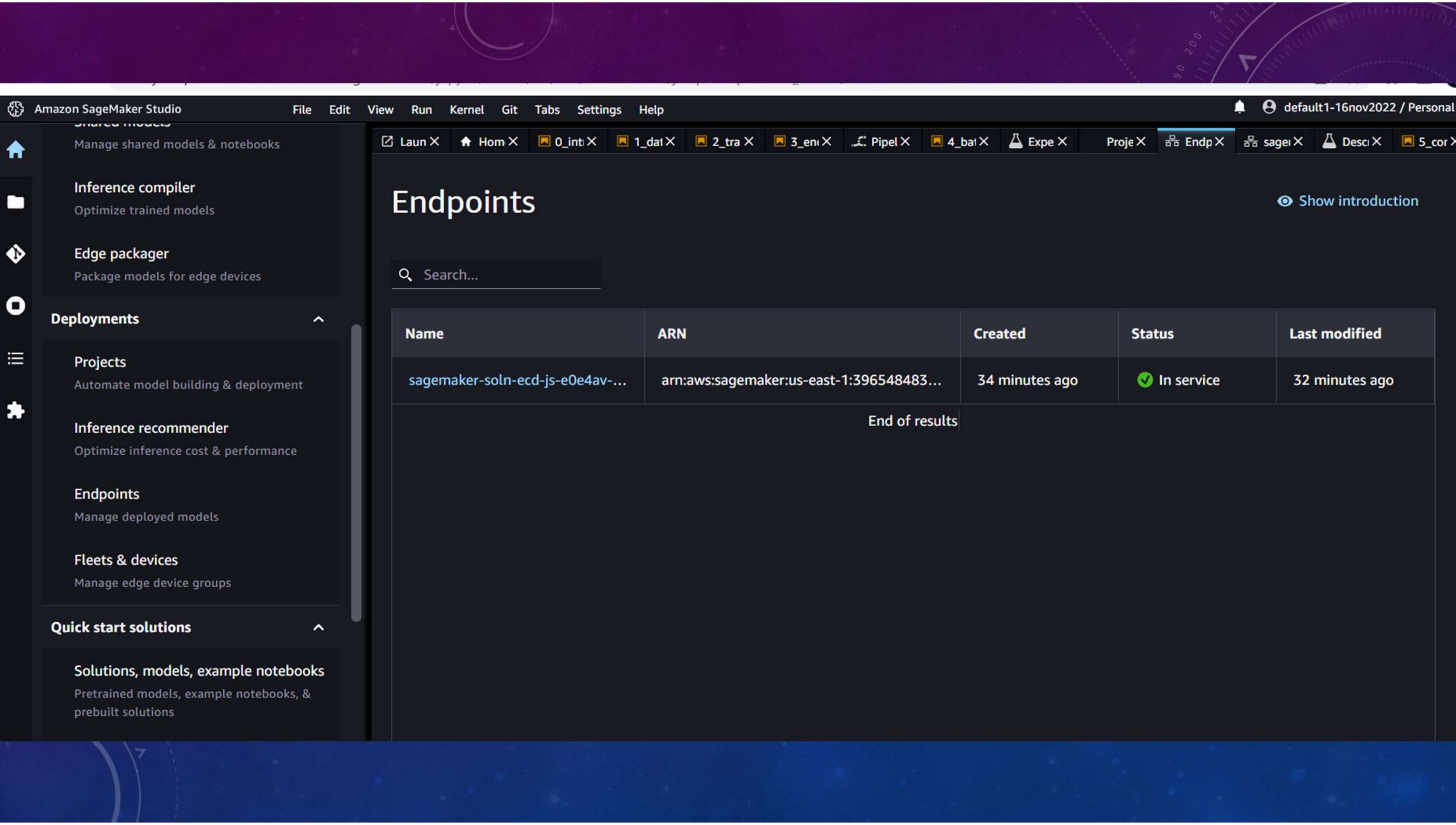
  - After this we see a number of features that increase the credit default risk:
    - A credit amount of 6000 EUR, a lack of employment and a credit duration of 36 months.
    - Another potential area for investigation, would be related to the repayment history feature.
    - Not having a very poor repayment history is associated with a higher credit default risk score.

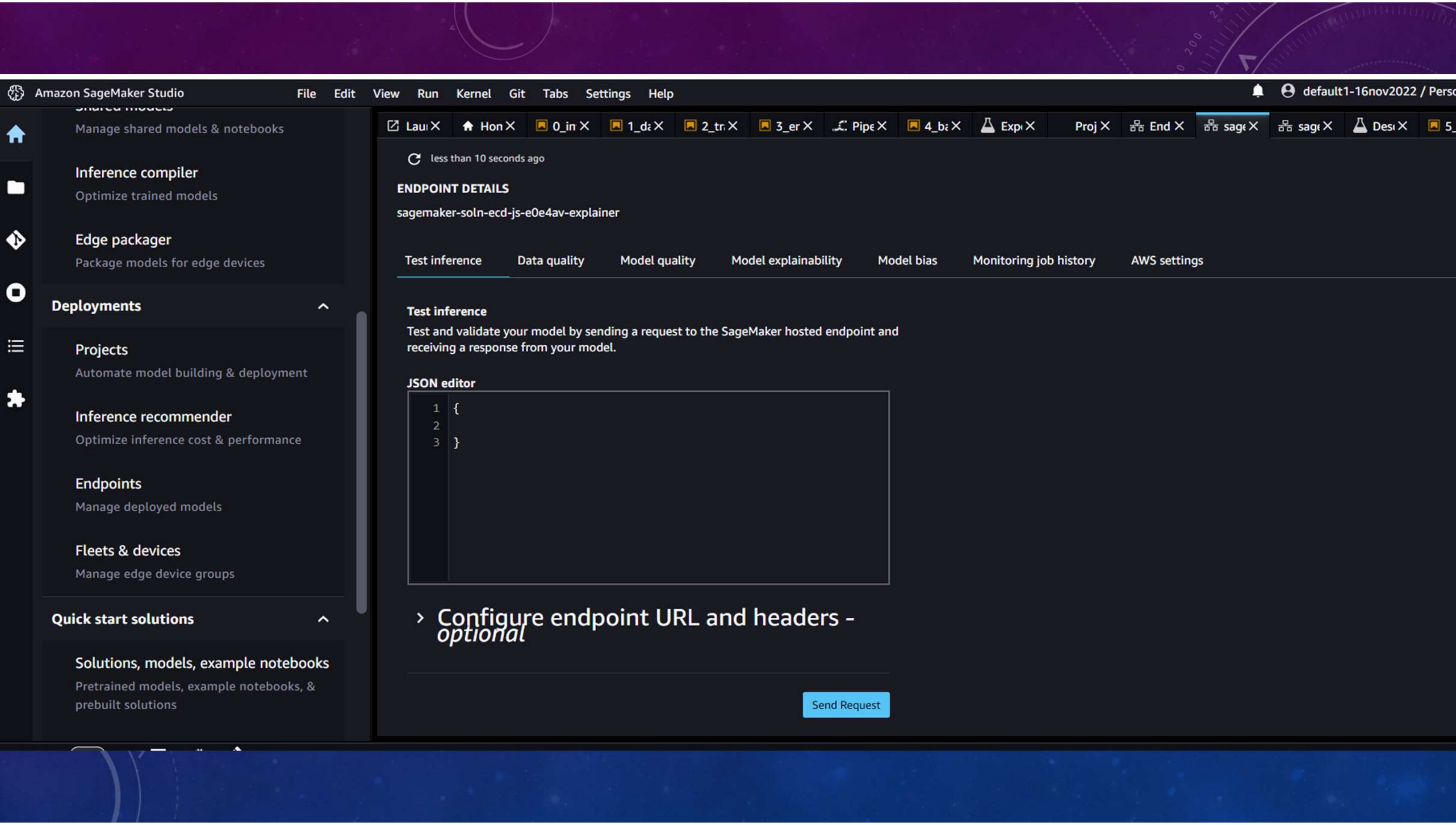
Counterfactual Example

```python
counter_sample = dict(sample)
counter_sample['finance__accounts__checking__balance'] = 'negative'  # from 'no_account'
counter_output = explainer.predict(counter_sample)
counter_explanation = visuals.detailed_explanation(counter_output)
visuals.WaterfallChart(
    baseline=counter_explanation['expected_value'],
    shap_values=counter_explanation['shap_values'],
    names=counter_explanation['feature_names'],
    feature_values=counter_explanation['feature_values'],
    descriptions=counter_explanation['feature_descriptions'],
    max_features=10,
    x_axis_label=x_axis_label,
).show()
```

Counterfactual Example

- Now let's switch the value of the checking account balance of the applicant from no account to negative.
- We can then see how the overall prediction of the model changes, and also see the updated contribution of this feature.
- Clearly, this application has become substantially more risky.


Amazon SageMaker Studio
File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help
default1-16nov2022 / Personal
Manage shared models & notebooks
Inference compiler
Optimize trained models
Edge packager
Package models for edge devices
Deployments
Projects
Automate model building & deployment
Inference recommender
Optimize inference cost & performance
Endpoints
Manage deployed models
Fleets & devices
Manage edge device groups
Quick start solutions
Solutions, models, example notebooks
Pretrained models, example notebooks, & prebuilt solutions
Endpoints
Show introduction
Search...
Name
ARN
Created
Status
Last modified
sagemaker-soln-ecd-js-e0e4av-...
arn:aws:sagemaker:us-east-1:396548483...
34 minutes ago
In service
32 minutes ago
End of results


Amazon SageMaker Studio
File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help
default1-16nov2022 / Perso
Manage shared models & notebooks
Inference compiler
Optimize trained models
Edge packager
Package models for edge devices
Deployments
Projects
Automate model building & deployment
Inference recommender
Optimize inference cost & performance
Endpoints
Manage deployed models
Fleets & devices
Manage edge device groups
Quick start solutions
Solutions, models, example notebooks
Pretrained models, example notebooks, & prebuilt solutions
less than 10 seconds ago
ENDPOINT DETAILS
sagemaker-soln-ecd-js-e0e4av-explainer
Test inference
Data quality
Model quality
Model explainability
Model bias
Monitoring job history
AWS settings
Test inference
Test and validate your model by sending a request to the SageMaker hosted endpoint and receiving a response from your model.
JSON editor
{
}
Configure endpoint URL and headers - optional
Send Request

DEMO

References

https://github.com/awslabs/sagemaker-explaining-credit-decisions

https://sagemaker-examples.readthedocs.io/en/latest/sagemaker_processing/fairness_and_explainability/fairness_and_explainability.html

My Github link : https://bit.ly/3BzAsQT

Linkedin : https://www.linkedin.com/in/sarbani-maiti-35b89111/

THANK YOU