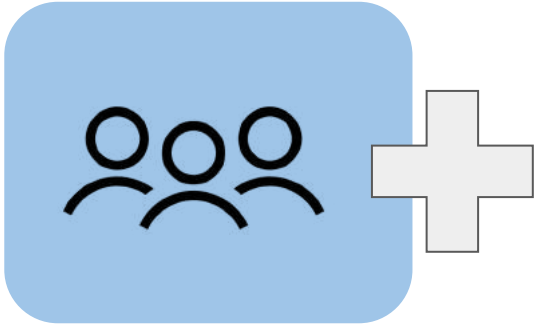
Copyright Notice

DeepLearning.AI

aws

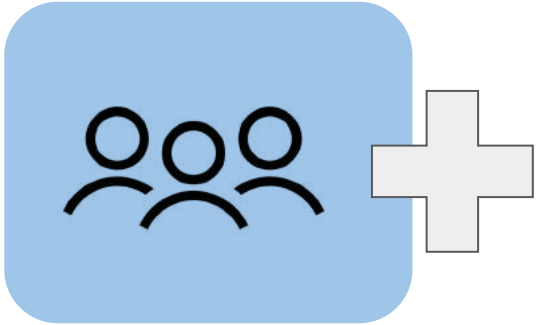
Data Labeling and Human-in-the-Loop Pipelines

Human & AI Collaboration



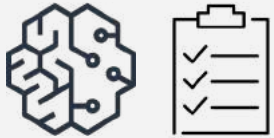
Ingest & Analyze	Prepare & Transform	Train & Tune	Deploy & Manage
Data exploration	Feature engineering	Automated ML	Model deployment
Bias detection	Feature store	Model train and tune	Automated pipelines
Amazon S3 & Amazon Athena	Amazon SageMaker Data Wrangler	Amazon SageMaker Autopilot	Amazon SageMaker Endpoints
AWS Glue	Amazon SageMaker Processing Jobs	Amazon SageMaker Training & Debugger	Amazon SageMaker Batch Transform
Amazon SageMaker Data Wrangler & Clarify	Amazon SageMaker Feature Store	Amazon SageMaker Hyperparameter Tuning	Amazon SageMaker Pipelines

Human & AI Collaboration



Data Labeling at scale

Human review of model predictions



Amazon SageMaker Ground Truth

for building data labeling workflows

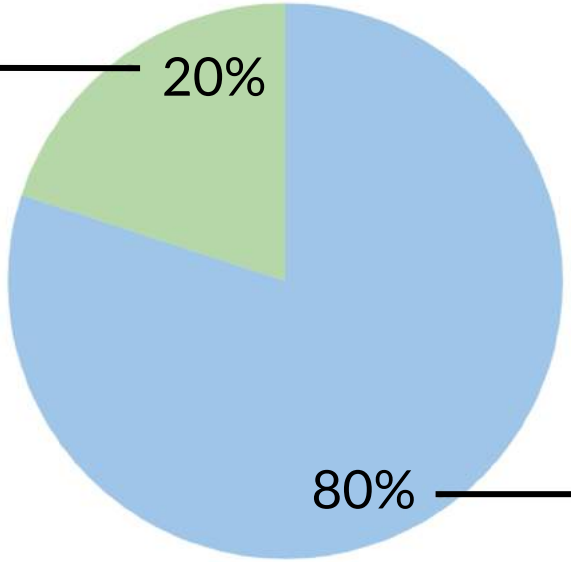
Amazon Augmented AI (Amazon A2I)

for implementing human in the loop pipelines.

Data Labeling

Preparing Data Takes Time

Time spent to develop, train and tune the model



Time spent on data preparation tasks

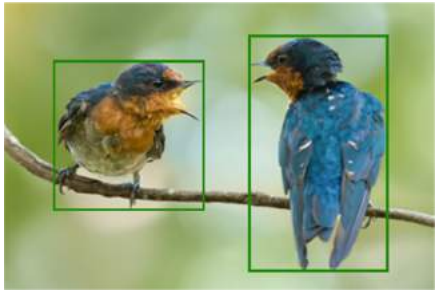
What Is Data Labeling?

- Process of identifying raw data and adding one or more meaningful and informative labels
- Labels provide context for machine learning models to learn from (supervised learning)
- Correctly labeled datasets are often called "ground truth"

Common Types Of Data Labeling

Image Data

Single-Label Classification



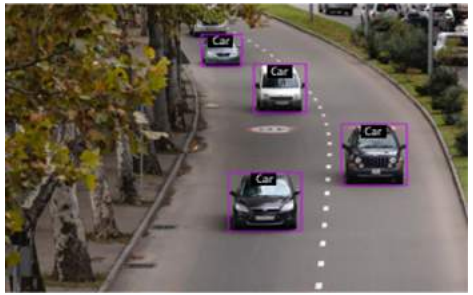
Bounding Box

Multi-Label Classification

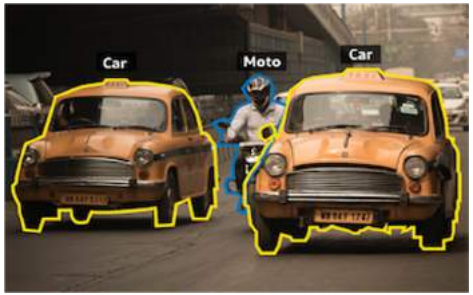
Semantic Segmentation

Video Data

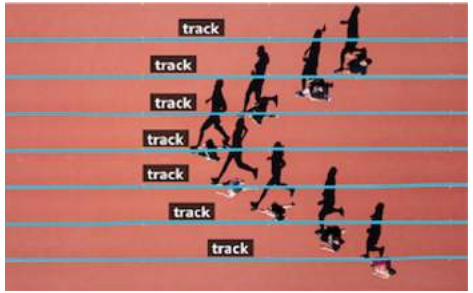
Classification



Bounding box



Polygon



Polyline

Key point

Text Data

Single-Label Classification

Multi-Label Classification

Named Entity Recognition

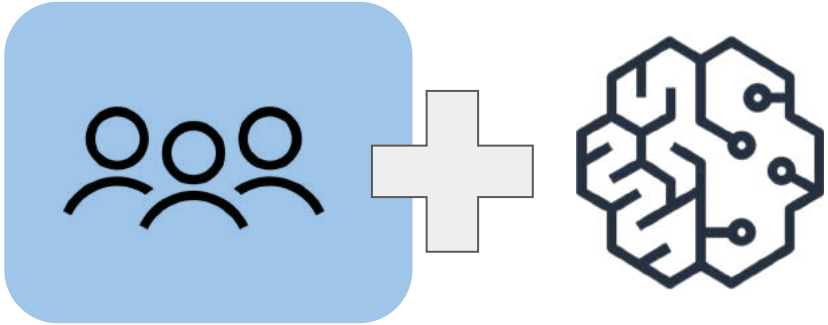
Challenges

Massive scale

High accuracy

Time consuming

How Can Data Labeling Be Done Efficiently?



- Access to additional human workforces
- Automated data labeling capabilities
- Assistive labeling features

Data Labeling

with Amazon SageMaker
Ground Truth

Amazon SageMaker Ground Truth

Input dataset

s3 BUCKET

Labeling Task

you define the labeling task and provide relevant labeling instructions. Ground Truth offers templates for common labeling tasks, where you need to click only a few choices and provide minimal instructions on how to get your data labeled. Alternatively, you can create your own custom template

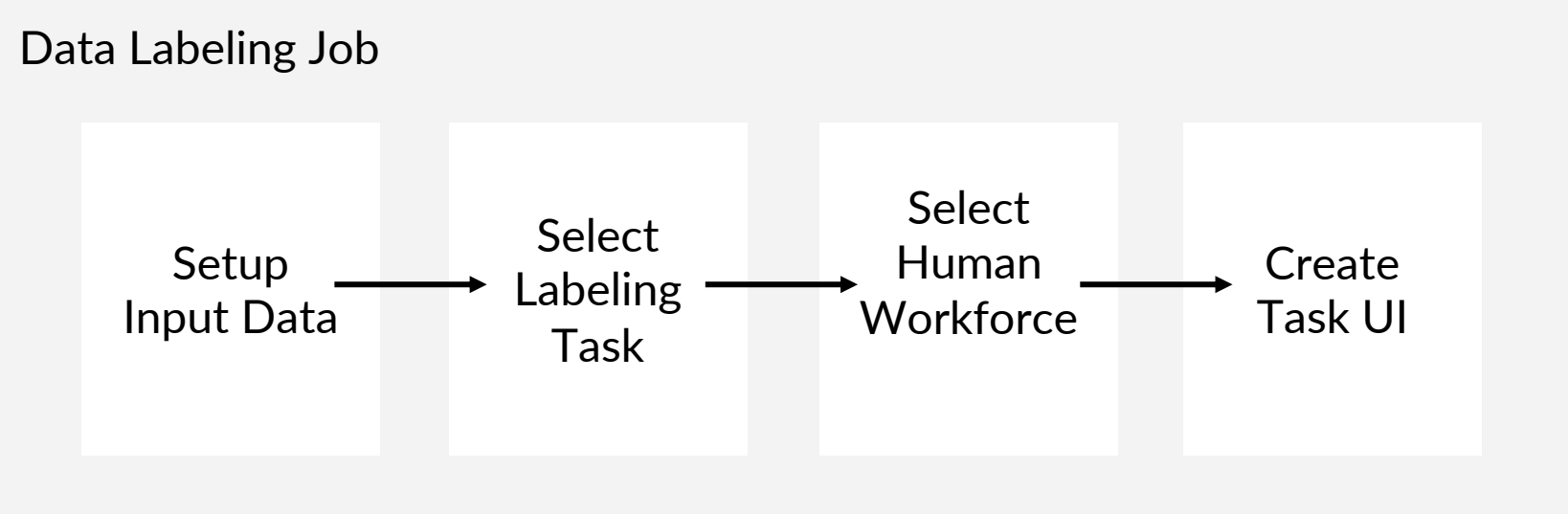
Amazon SageMaker Ground Truth

Labeled dataset

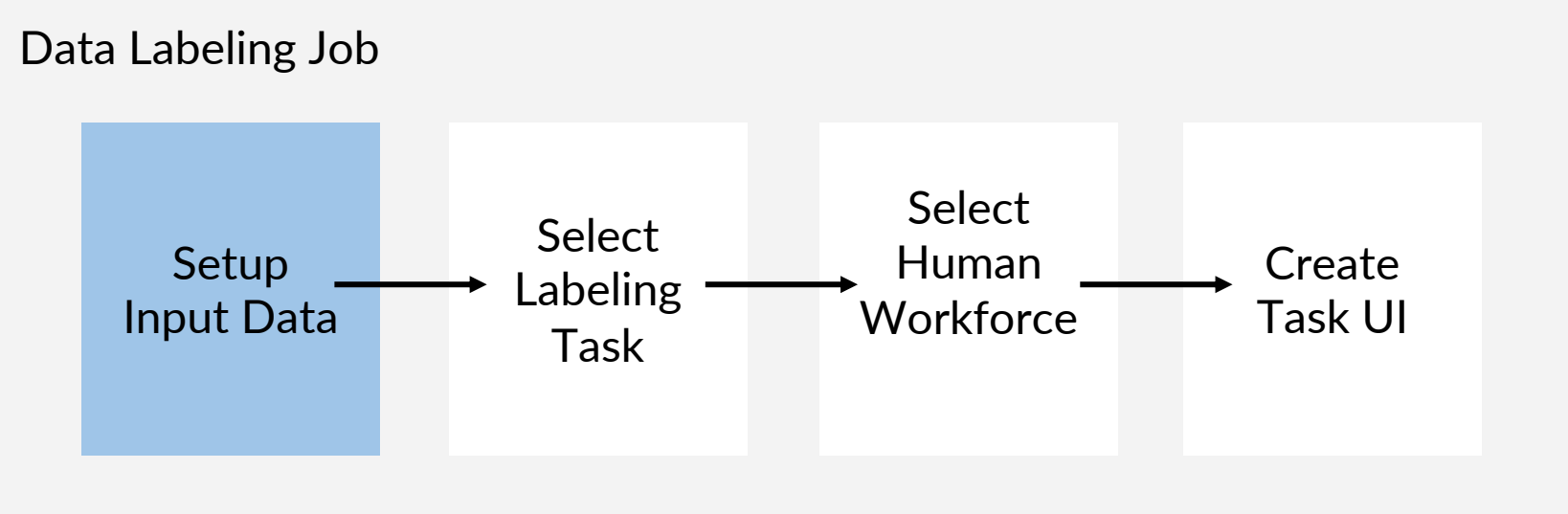
Human Workforce

a crowdsourced Amazon Mechanical Turk workforce of over 500,000 labelers, a private team of your co-workers, or one of the third-party data labeling service providers listed on the AWS marketplace

Steps


Data Labeling Job
Setup Input Data
Select Labeling Task
Select Human Workforce
Create Task UI

Steps


Data Labeling Job
Setup Input Data
Select Labeling Task
Select Human Workforce
Create Task UI

Setup Input Data

- Automated data setup
  - Provide the S3 location of the dataset you want labeled

Setup Input Data

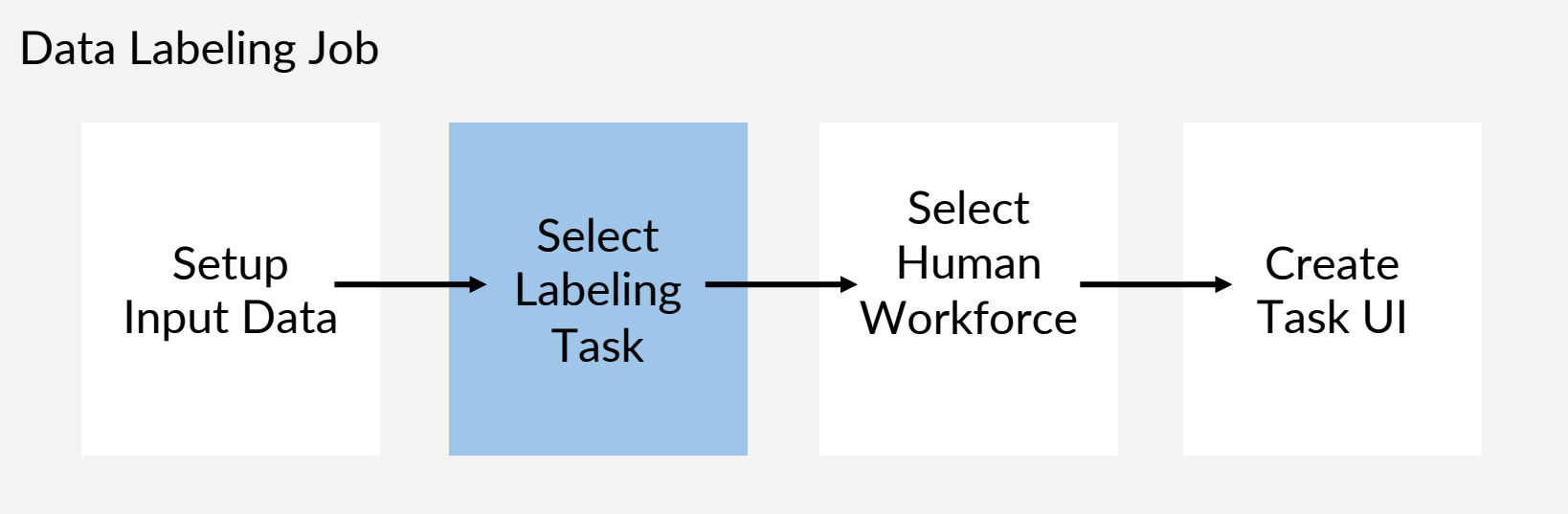
- Automated data setup
  - Provide the S3 location of the dataset you want labeled

- Manual data setup
  - Provide the S3 location of a file (an input manifest file) that identifies the data objects you want labeled

```
{"source-ref": "S3 bucket location 1"}
   ...
{"source-ref": "S3 bucket location n"}
```

```
{"source": "I love this product"}
   ...
{"source": "It is ok"}
```

Steps


Data Labeling Job
Setup Input Data
Select Labeling Task
Select Human Workforce
Create Task UI

Select Labeling Task

- Image Data
- Video Data
- Text Data
- Custom

Select Labeling Task

- Image Data
- Video Data
- Text Data
- Custom

Image Classification (Single Label)

Image Classification (Multi-Label)

Bounding Box

Semantic Segmentation

Label Verification

Select Labeling Task

- Image Data
- Video Data
- Text Data
- Custom

Video Clip Classification

Video Object Detection/Tracking

Bounding Box
Polygon
Polyline
Key Point

Select Labeling Task

- Image Data
- Video Data
- Text Data
- Custom

Text Classification (Single Label)

Text Classification (Multi-Label)

Named Entity Recognition

Select Labeling Task

- Image Data
- Video Data
- Text Data
- Custom

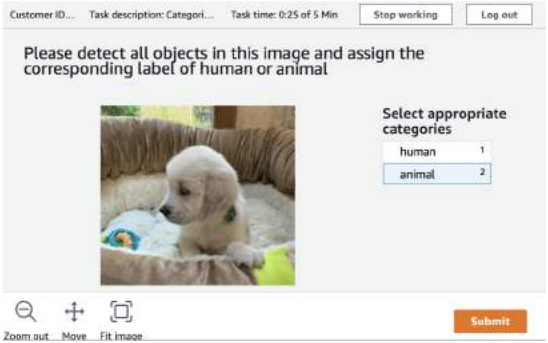
Your custom labeling task

Custom Labeling Task

AWS Lambda

Pre-annotation Lambda Function
Pre-process each data object

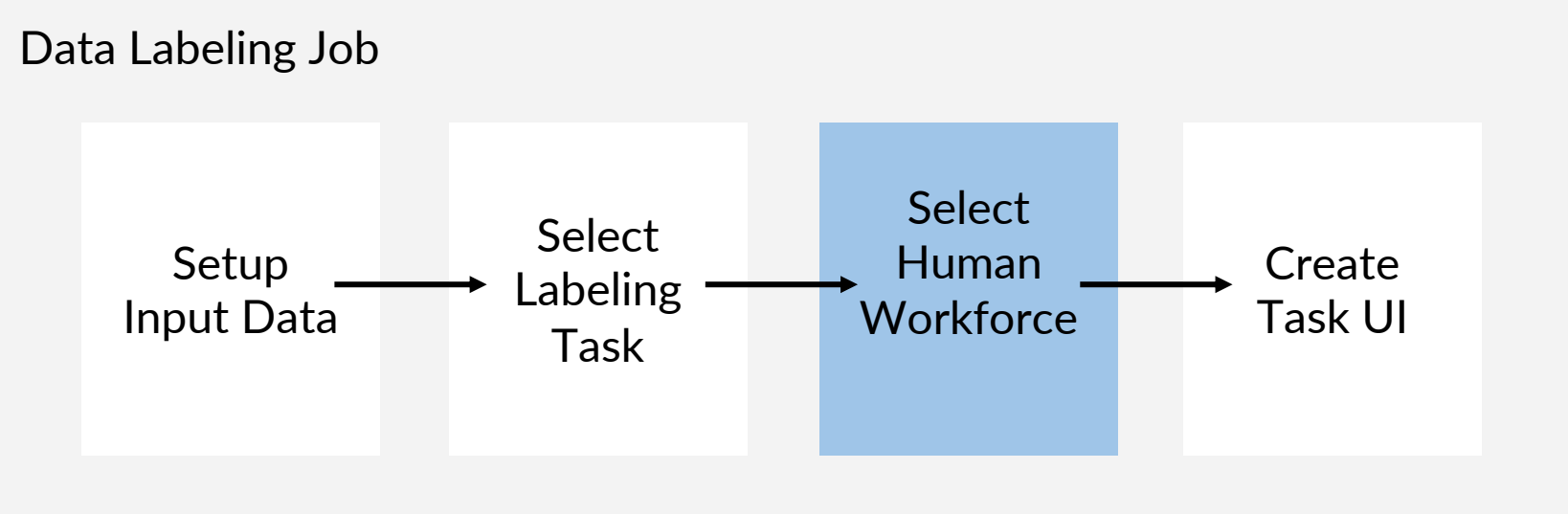
Amazon SageMaker
Ground Truth



AWS Lambda

Post-annotation Lambda Function
Process each labeling result
Consolidate annotations

Steps


Data Labeling Job
Setup Input Data
Select Labeling Task
Select Human Workforce
Create Task UI

Human Workforce Options

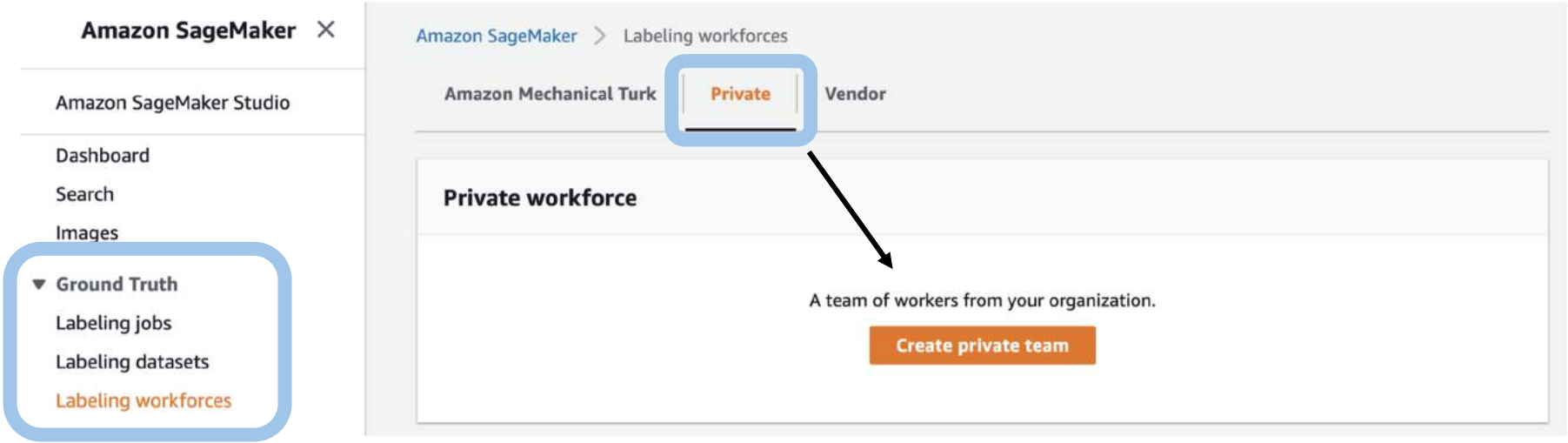
Amazon Mechanical Turk

Private

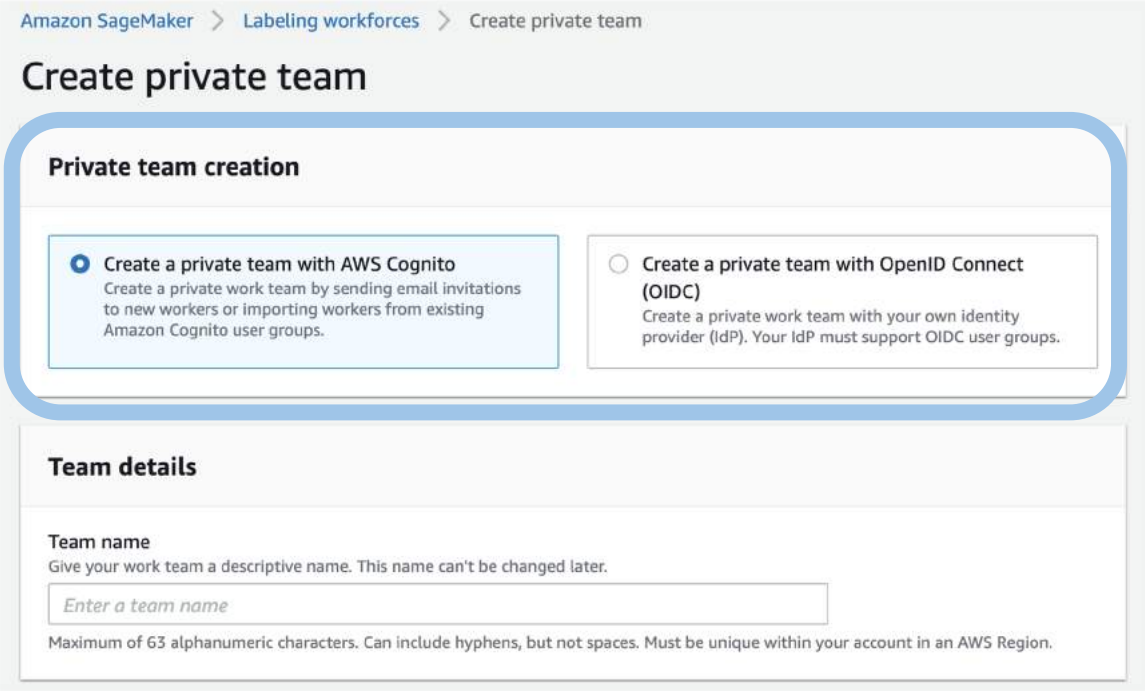
Vendor

You can restrict worker access to tasks to allowable IP addresses

Setup Private Workforce (UI)

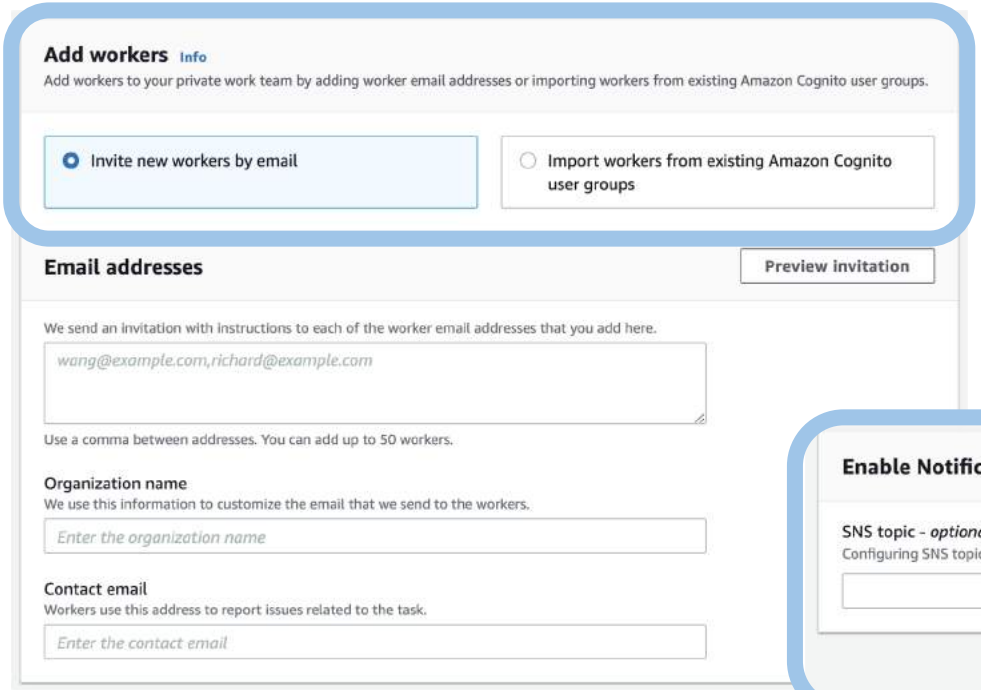


Setup Private Workforce (UI)

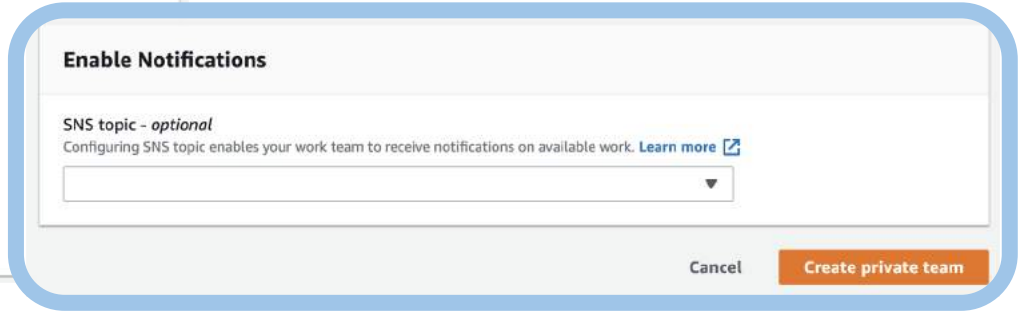


- User authentication via Amazon Cognito or OpenID Connect (OIDC)
- You can connect Amazon Cognito to your enterprise identity provider

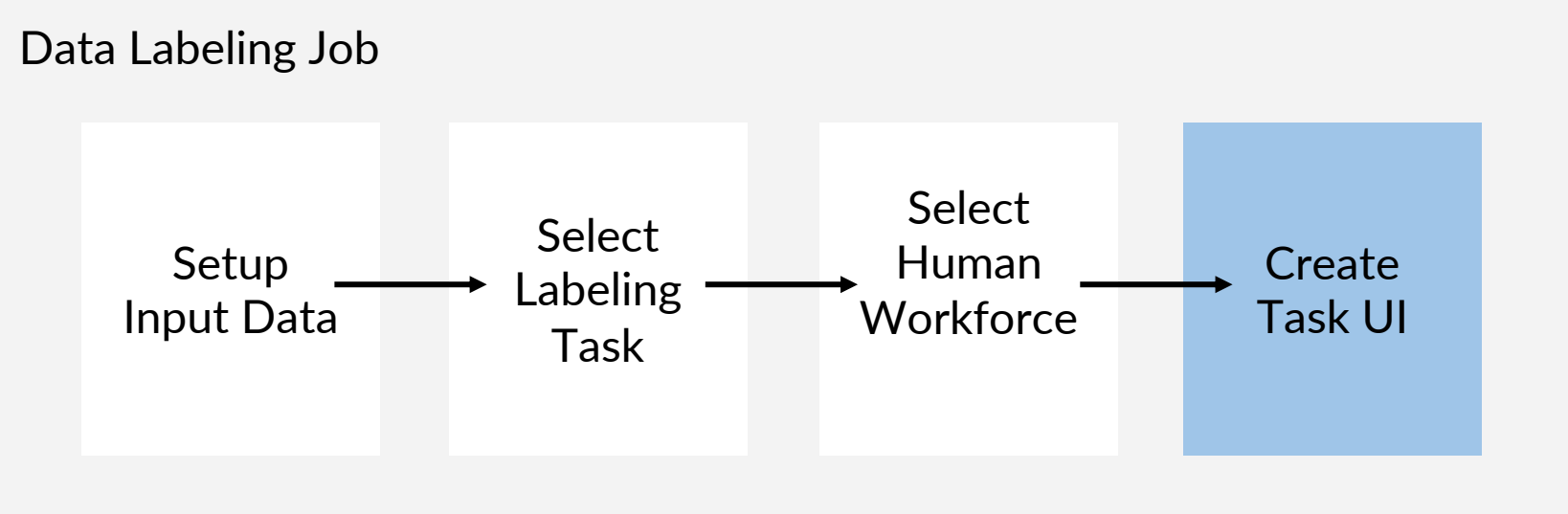
Setup Private Workforce (UI)



- Invite workers by email
- Enable notifications for work team



Steps


Data Labeling Job
Setup Input Data
Select Labeling Task
Select Human Workforce
Create Task UI

Human Task UI

- Combination of HTML, CSS, JavaScript, the Liquid templating language, and Crowd HTML Elements.
- Use built-in templates, or define custom task UI

```
<script src="https://assets.crowd.aws/crowd-html-elements.js"></script>

<crowd-form>
          ...
</crowd-form>
```

Sample Task UI For Text Classification

```
<script src="https://assets.crowd.aws/crowd-html-elements.js"></script>
<crowd-form>
    <crowd-classifier name="sentiment"
                      categories="['-1', '0', '1']"
                      initial-value="{{ task.input.initialValue }}"
                      header="Classify Reviews into Sentiment:  -1 (negative),
                                                                 0 (neutral), and
1 (positive)">

        <classification-target>
            {{ task.input.taskObject }}
        </classification-target>
    ...
```

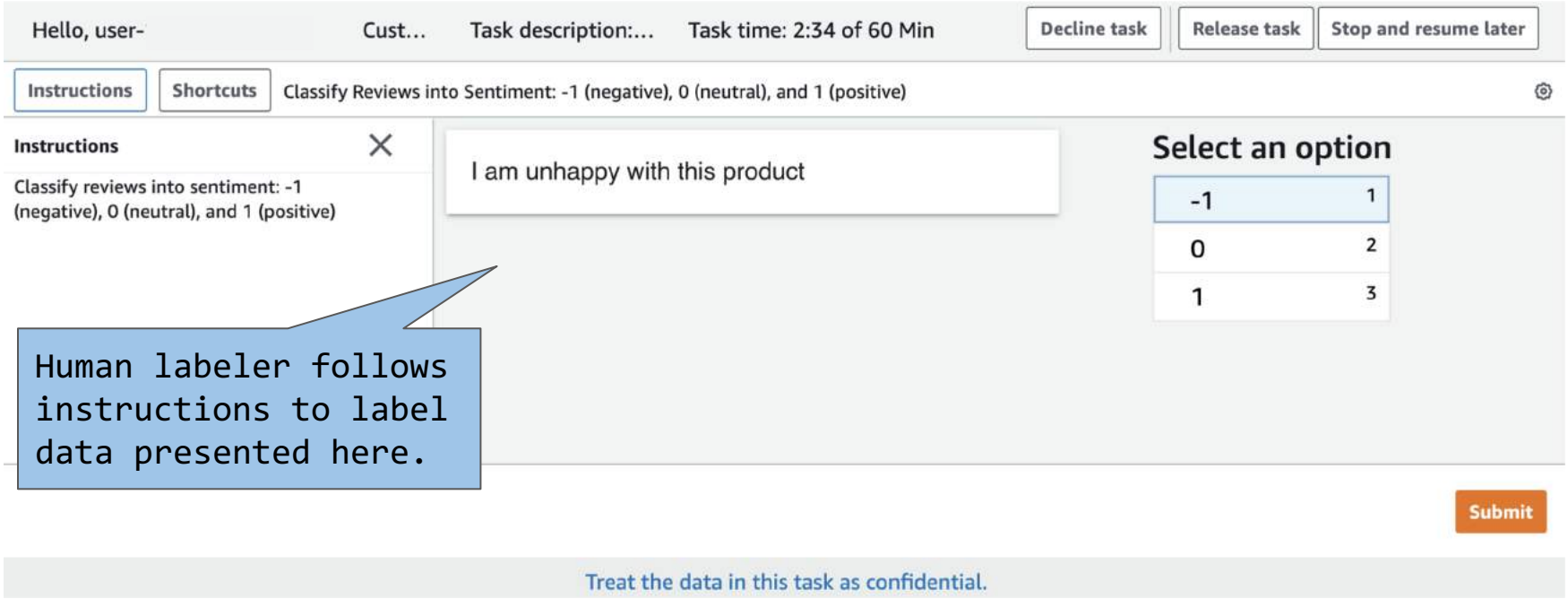
Customize instructions to match product reviews text classification

Sample Task UI For Text Classification

```
    ...
        <full-instructions header="Classify reviews into sentiment:
                                                                -1 (negative), 0
(neutral), and 1 (positive)">
            <p><strong>1</strong>: joy, excitement, delight</p>
            <p><strong>0</strong>: neither positive or negative</p>
            <p><strong>-1</strong>: anger, sarcasm, anxiety</p>
        </full-instructions>
        <short-instructions>
            Classify reviews into sentiment:
                      -1 (negative), 0 (neutral), and 1 (positive)
        </short-instructions>
    </crowd-classifier></crowd-form>
```

Provide more detailed instructions.

Define Task UI For Text Classification



Human labeler follows instructions to label data presented here.

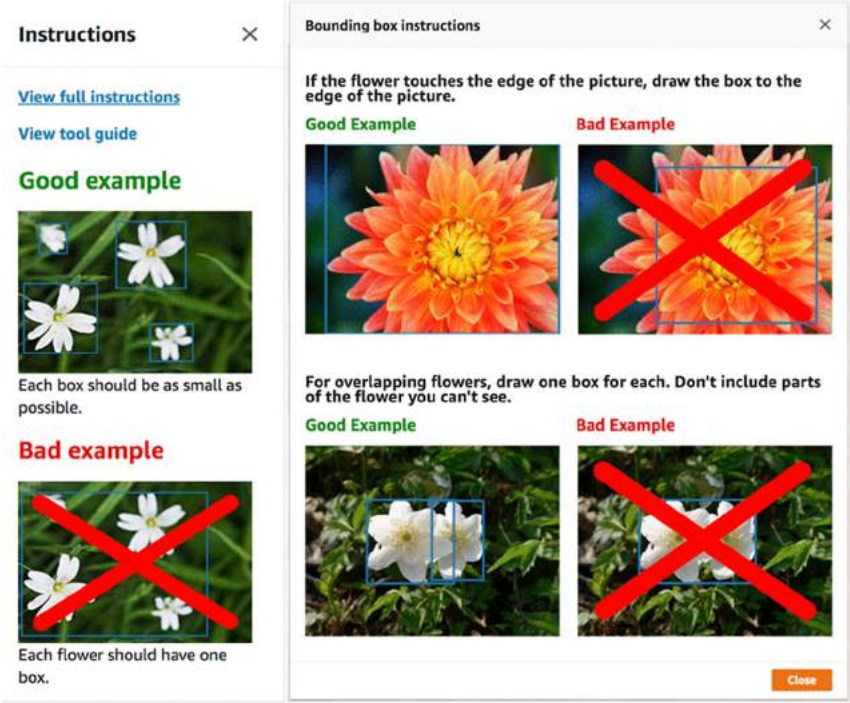
Data Labeling

Best Practices

Best Practices

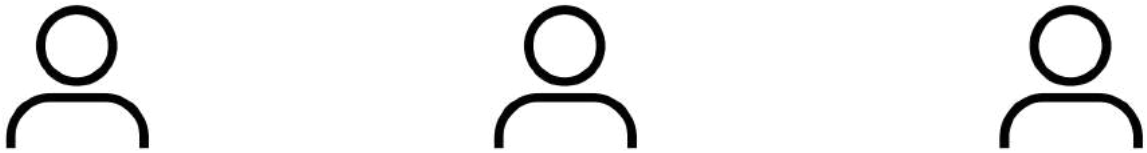
- Provide clear instructions
- Consolidate annotations to improve label quality
- Verify and adjust labels
- Use automated data labeling on large datasets (Active Learning)
- Re-use prior labeling jobs to create hierarchical labels

Provide Clear Instructions



- Provide examples of good and bad annotations
- Only show relevant labels
- Use multiple workers per task

Consolidate Annotations To Improve Label Quality



Labeler A

Labeler B

Labeler C

min. 3-5

"I like this shirt." > positive

"I like this shirt." > positive

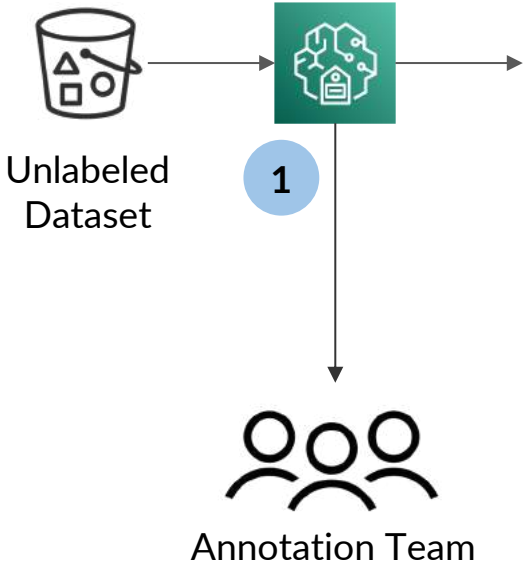
"I like this shirt." > negative

"I like this shirt." > positive

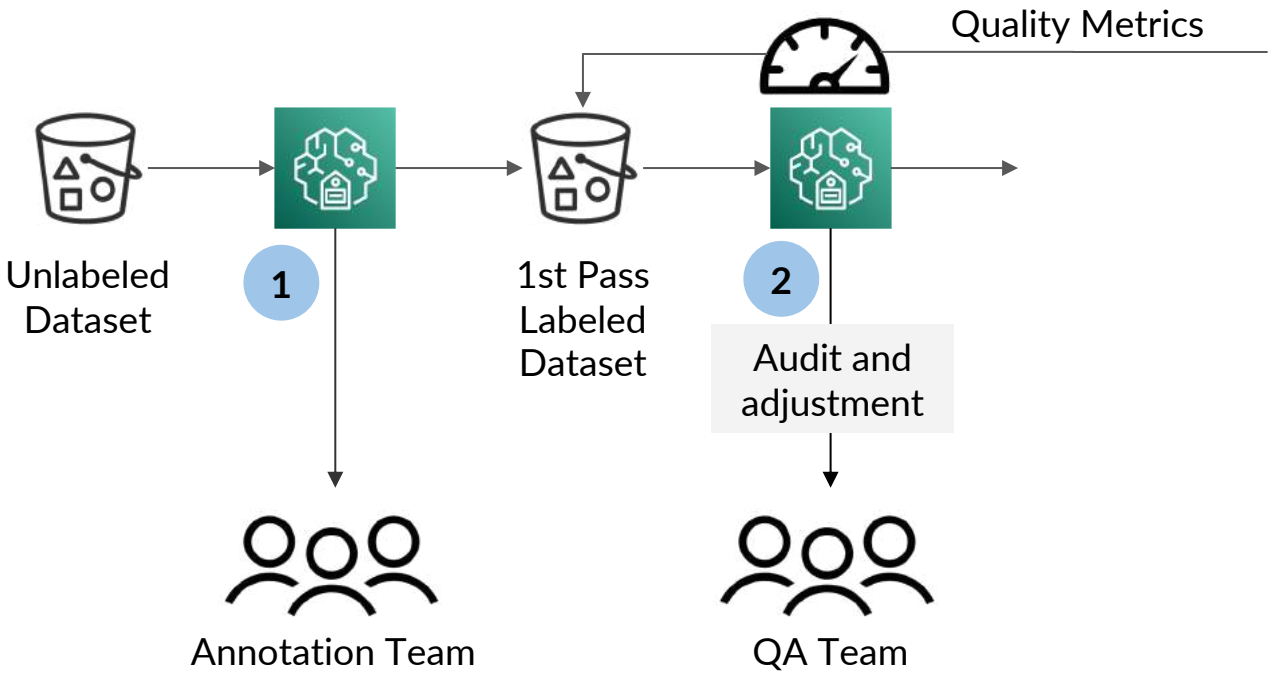
- Assess similarity
- Assess the most likely label

Use built-in annotation consolidation functions (or provide a custom function)

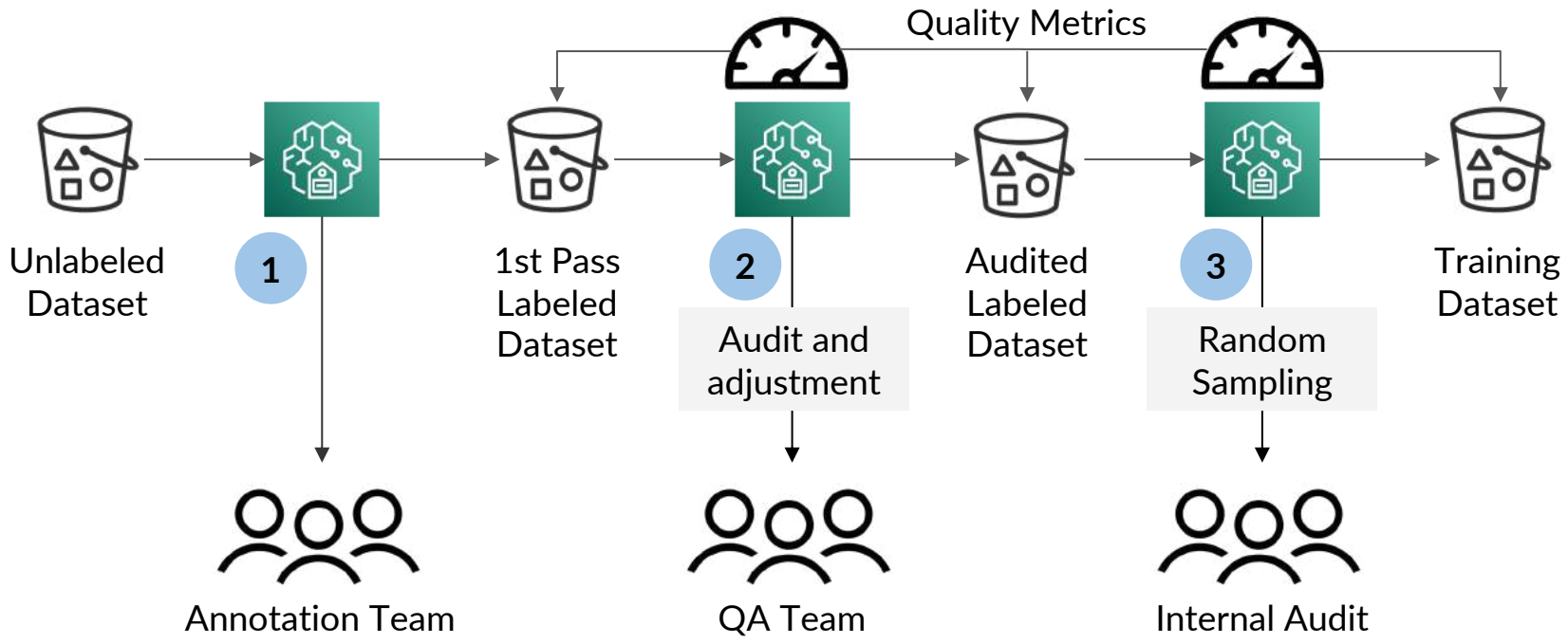
Verify And Adjust Labels



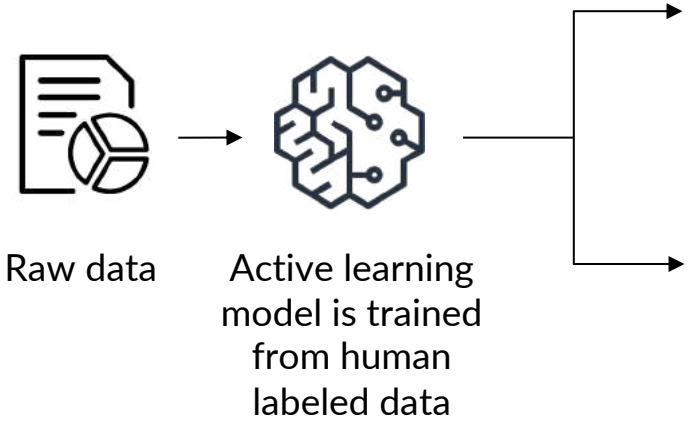
Verify And Adjust Labels



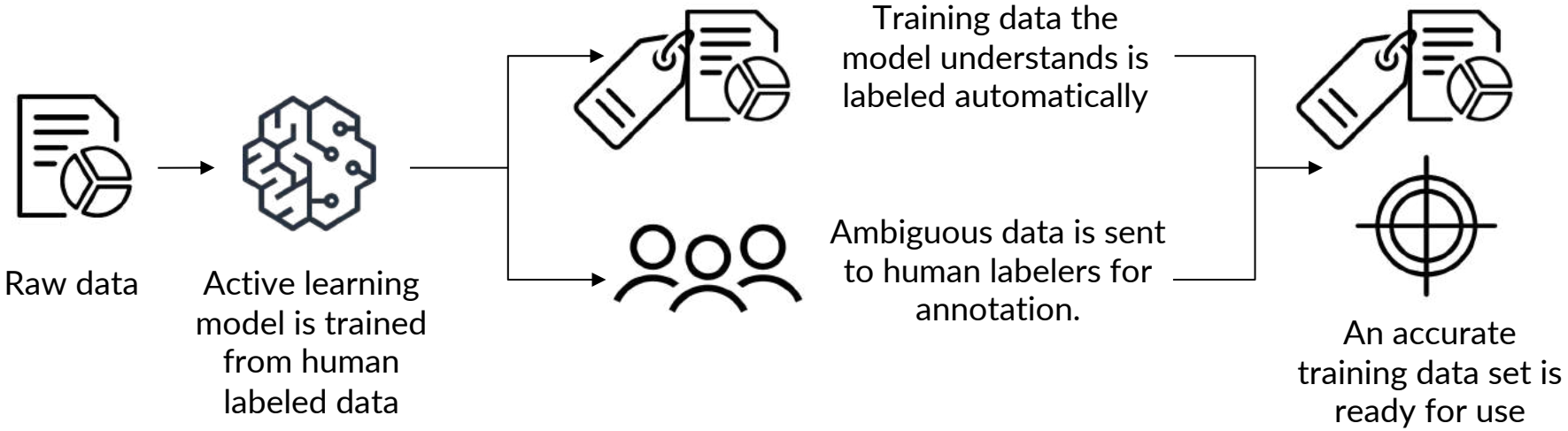
Verify And Adjust Labels



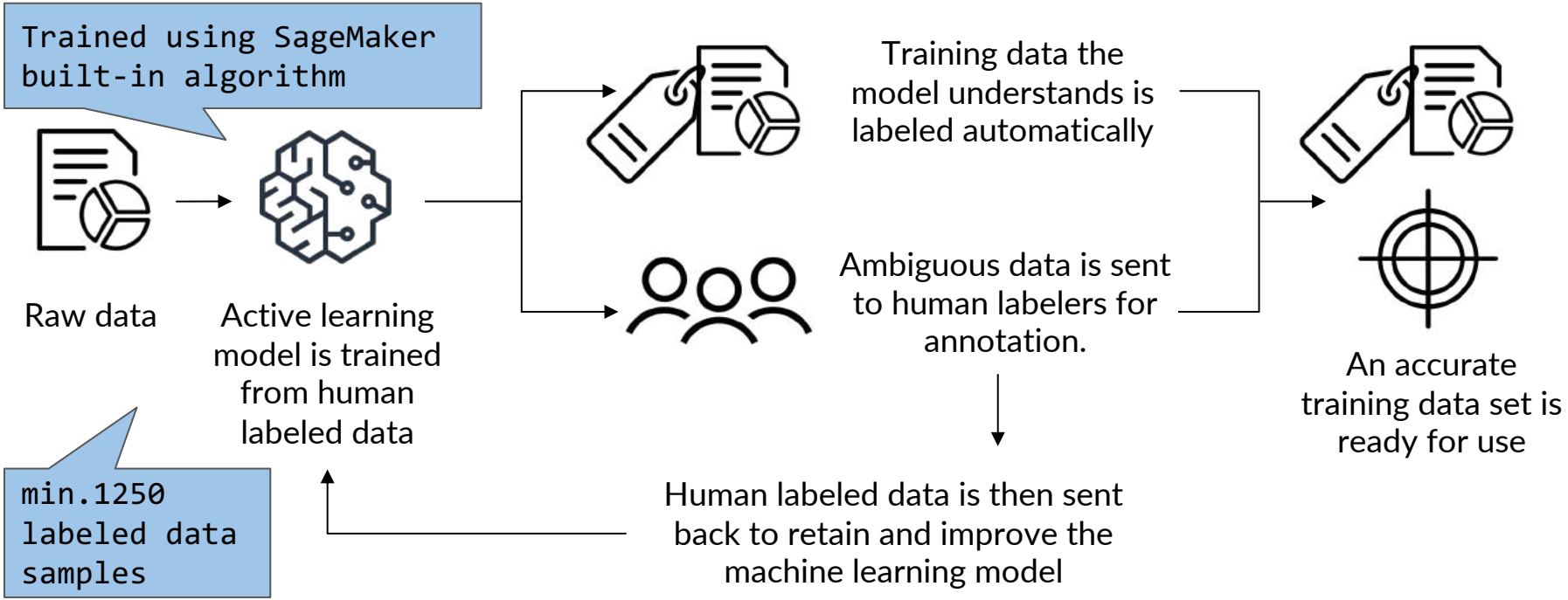
Use Automated Data Labeling On Large Datasets



Use Automated Data Labeling On Large Datasets



Use Automated Data Labeling On Large Datasets



but it is better to have 5,000 objects

Re-Use Prior Labeling Jobs To Create Hierarchical Labels

Labeling Job 1:
Identify cat or dog

Filter
"dog"

Labeling Job 2:
Identify dog breed

Label:

"dog"

Label:

"dog"
"english_retriever"

a first data labeling job could classify objects in an image into cats and dogs, and in a second labeling jobs, you could filter the images that contain a dog and that an additional label for the specific dog breed. The result of the second labeling job is an augmented manifest file. Augmented refers to the fact that the manifest file contains the labels from both labeling jobs.

Human-In-The-Loop Pipelines

Image Classification

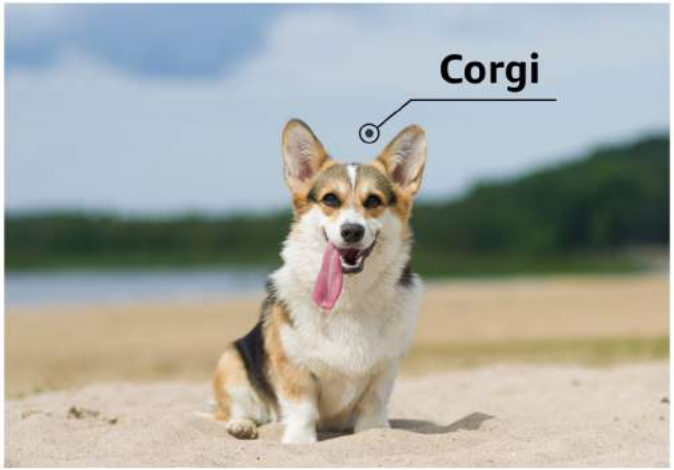
Task: Verify the appropriate dog breed has been classified in the image

1. Read the task carefully and inspect the image.
2. Read the options and review the examples to understand more about the dog breed categories.
3. Choose the appropriate dog breed category that best suits the image.

Examples

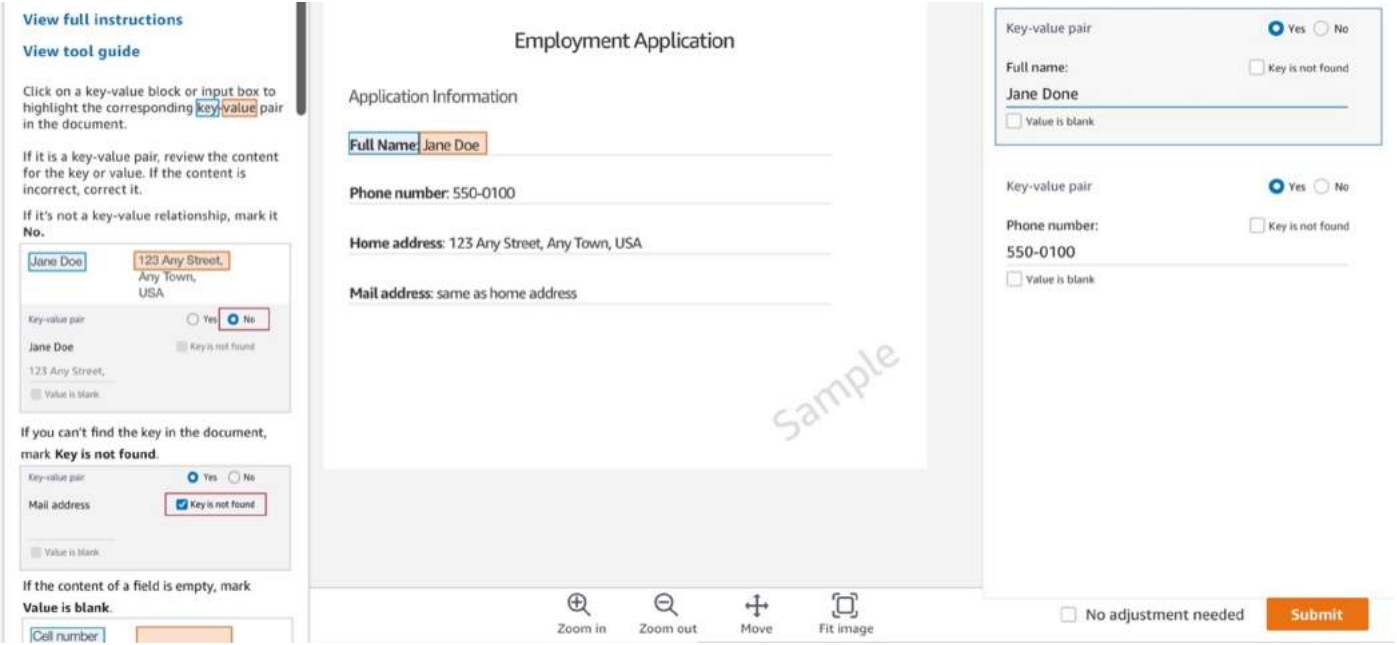
Corgi

Chihuahua

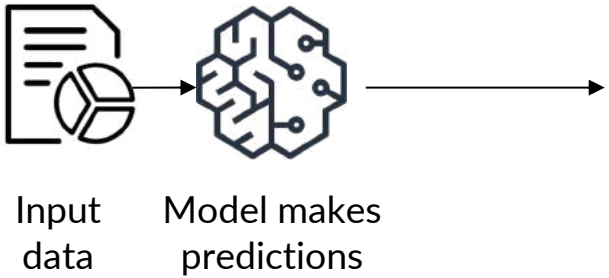


Form Extraction

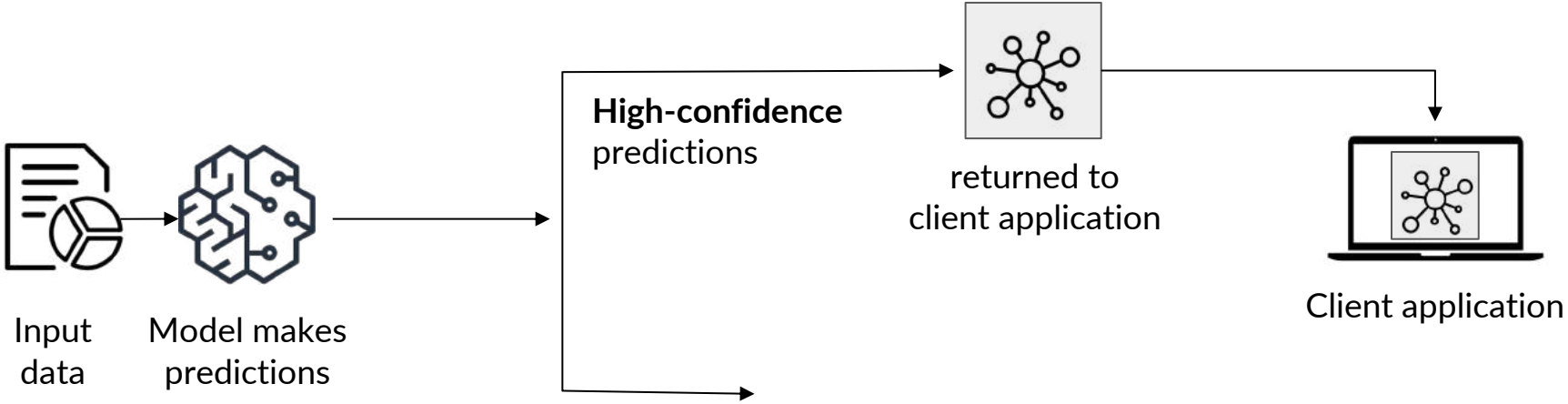
Task: Review the key-value pairs to the right and correct them if they do not match the document



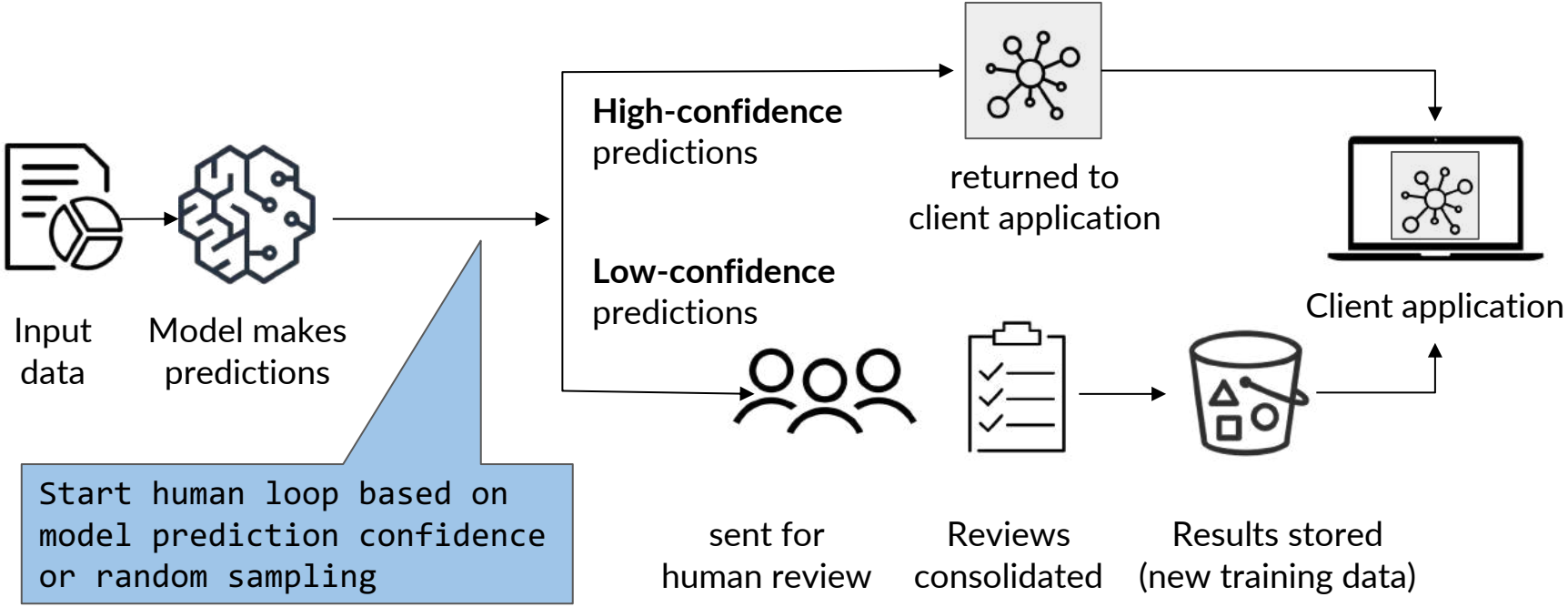
Implement Human Review Of Model Predictions



Implement Human Review Of Model Predictions



Implement Human Review Of Model Predictions



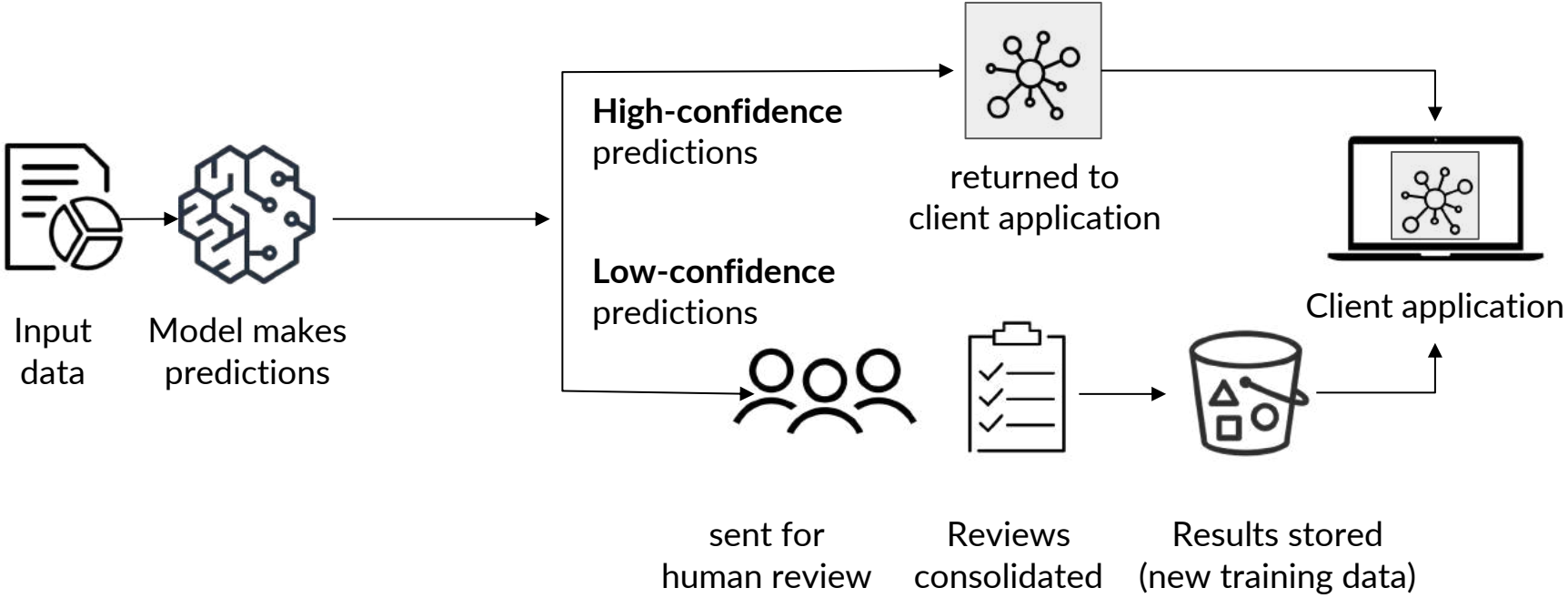
Challenges

- Need ML scientists, engineering, and operations teams
- Need to manage large number of reviewers
- Need to write custom software to manage review tasks
- Difficult to achieve high review accuracy

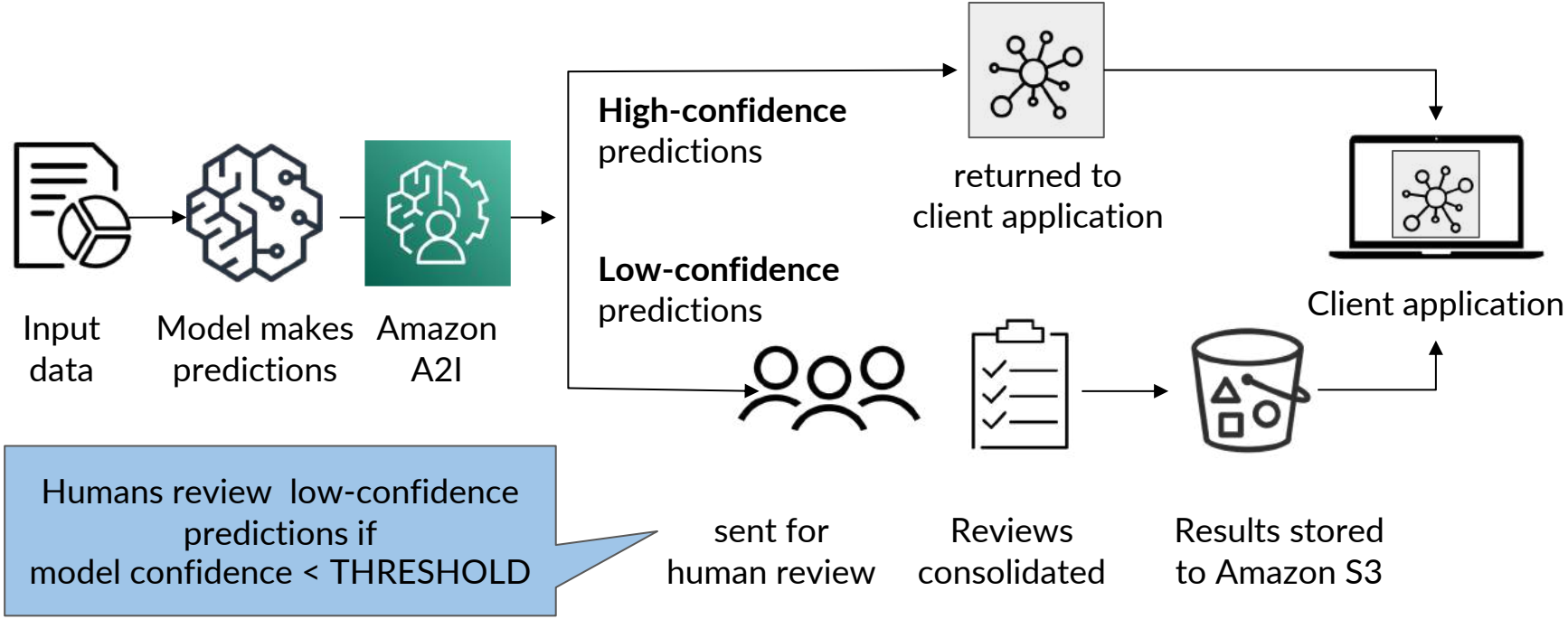
Human-In-The-Loop Pipelines

with Amazon Augmented AI (Amazon A2I)

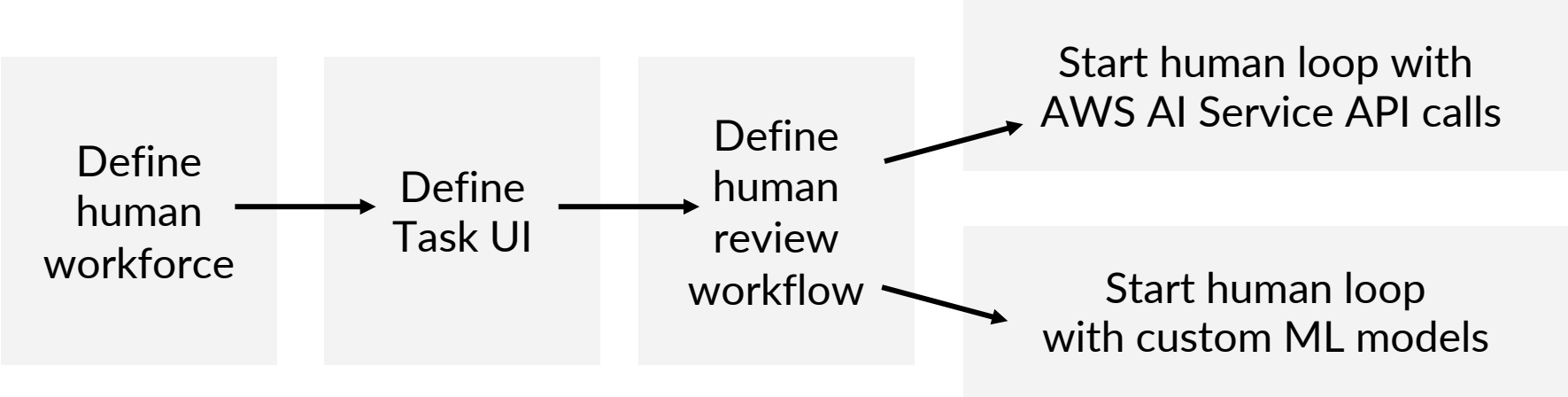
Implement Human Review Of Model Predictions



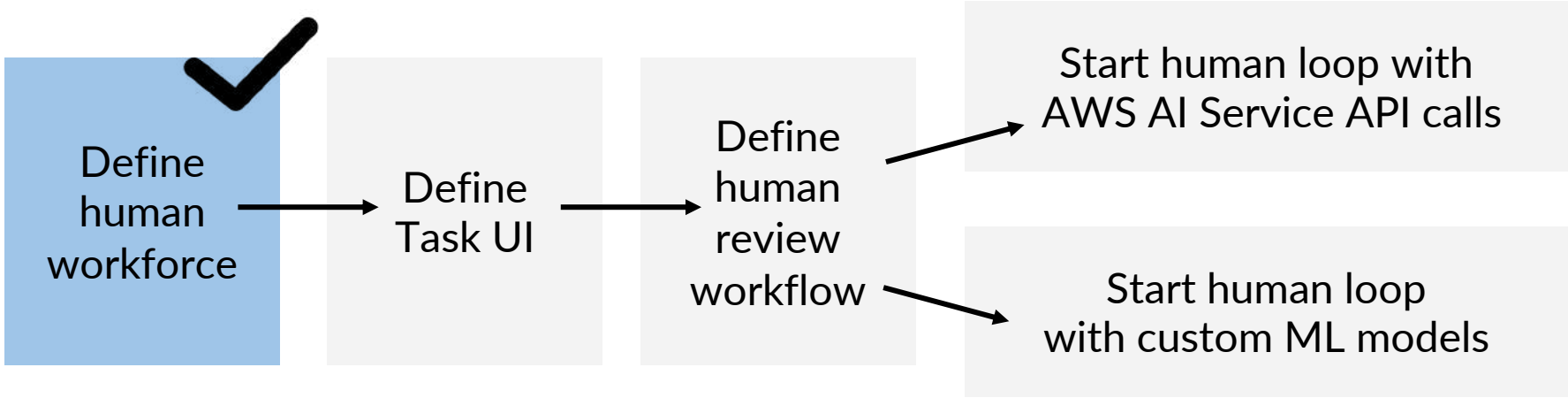
Amazon A2I


Input data
Model makes predictions
Amazon A2I
High-confidence predictions
returned to client application
Client application
Low-confidence predictions
sent for human review
Reviews consolidated
Results stored to Amazon S3
Humans review low-confidence predictions if model confidence < THRESHOLD

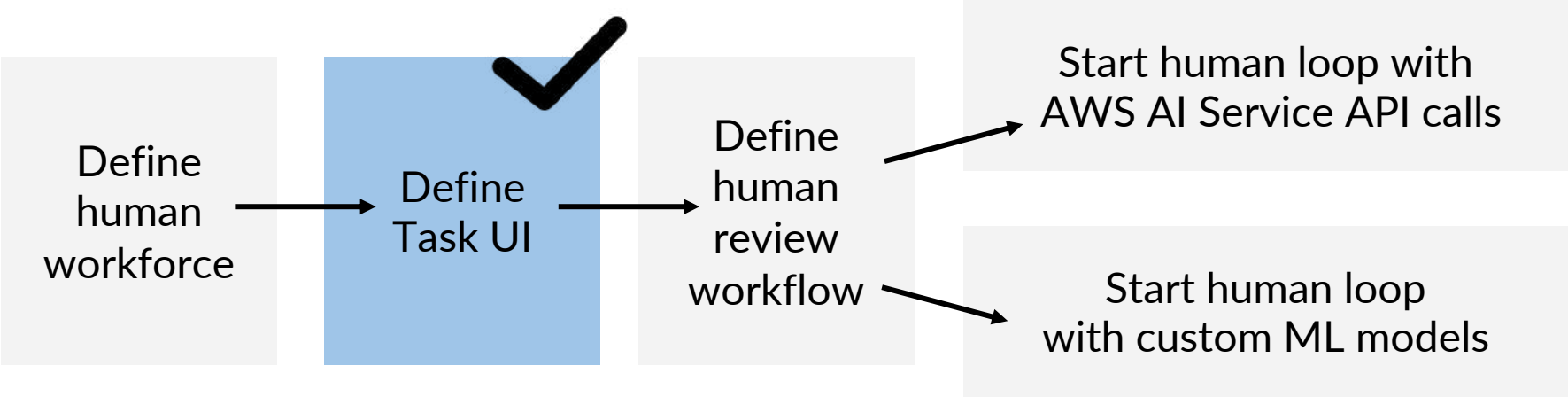
Steps


Define human workforce
Define Task UI
Define human review workflow
Start human loop with AWS AI Service API calls
Start human loop with custom ML models

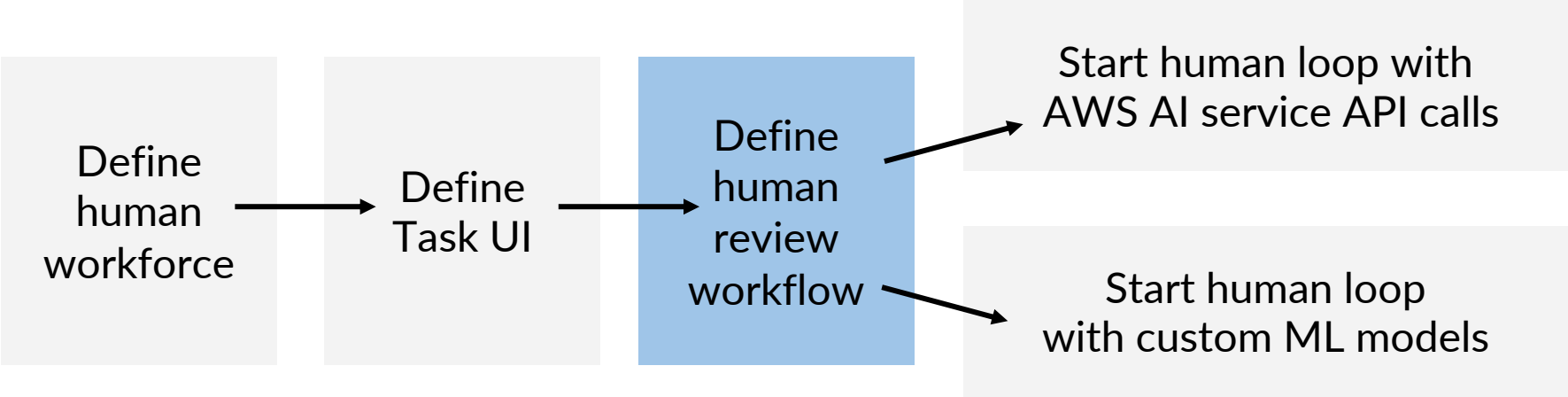
Steps


Define human workforce
Define Task UI
Define human review workflow
Start human loop with AWS AI Service API calls
Start human loop with custom ML models

Steps


Define human workforce
Define Task UI
Define human review workflow
Start human loop with AWS AI Service API calls
Start human loop with custom ML models

Steps


Define human workforce
Define Task UI
Define human review workflow
Start human loop with AWS AI service API calls
Start human loop with custom ML models

Define Human Review Workflow

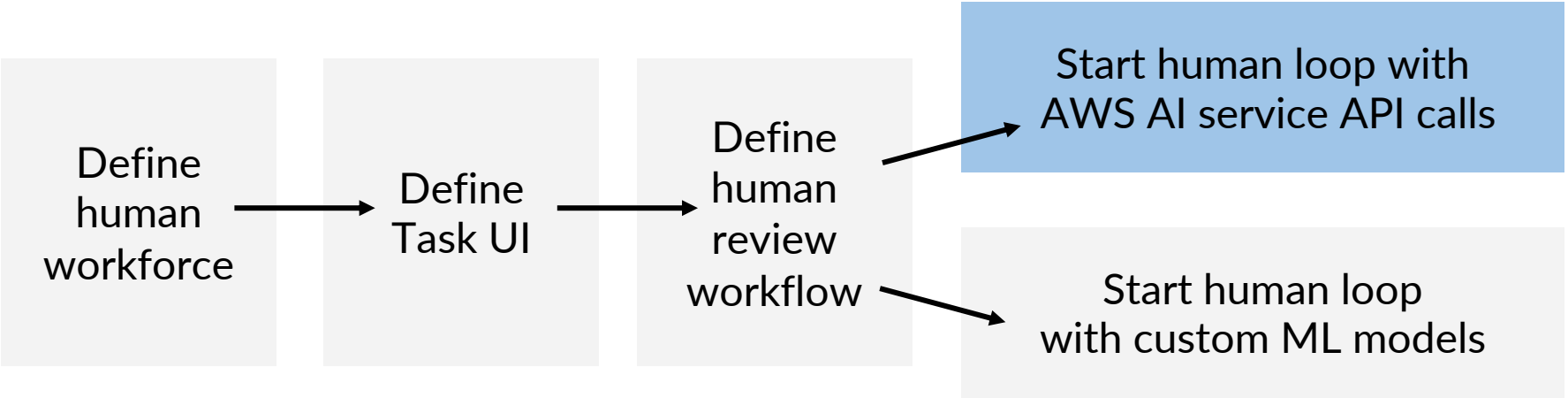
```
create_workflow_definition_response = sm.create_flow_definition(
    FlowDefinitionName=<NAME>,
    RoleArn=role,
    HumanLoopConfig={
        "WorkteamArn": ...,
        "HumanTaskUiArn": ...,
        "TaskCount": 1,
        "TaskDescription": "Classify Reviews into sentiment:  -1 (negative),
                                                    0 (neutral), 1
(positive)",
        "TaskTitle": ... },
    OutputConfig={"S3OutputPath": output_path}, ... )

augmented_ai_flow_definition_arn =
         create_workflow_definition_response["FlowDefinitionArn"]
```

Workforce and team to use

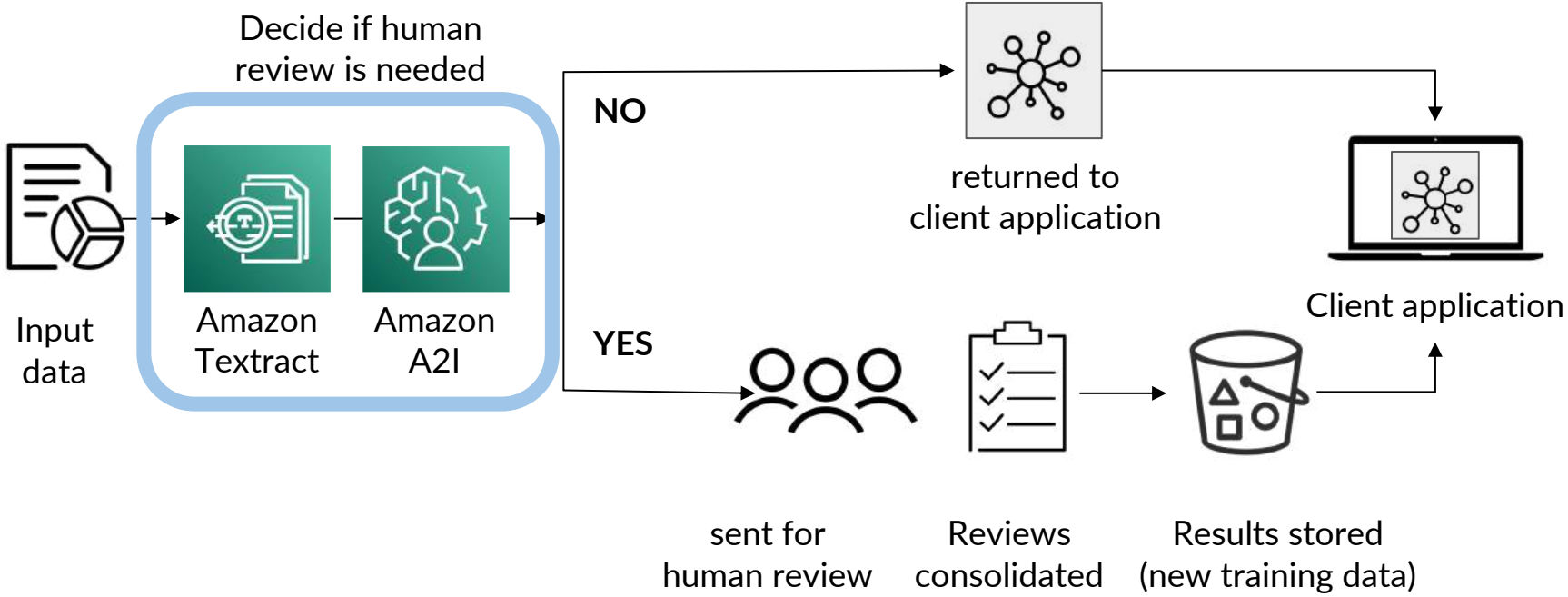
Human Task UI and additional configuration

Where to store output data

Steps


Define human workforce
Define Task UI
Define human review workflow
Start human loop with AWS AI service API calls
Start human loop with custom ML models

Start Human Loop With Amazon Textract

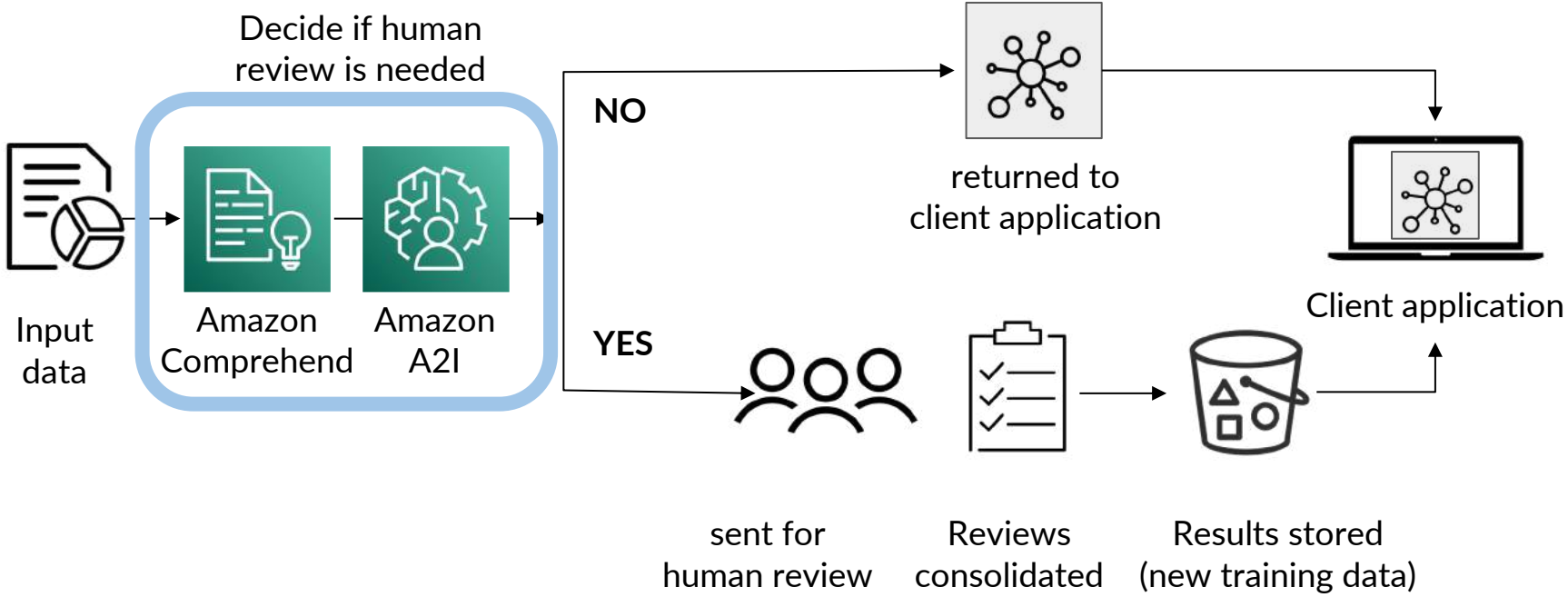


Start Human Loop With Amazon Textract

```
def analyze_document_with_a2i(document_name, bucket):
    response = textract.analyze_document(
        Document={'S3Object': {'Bucket': bucket, 'Name': document_name}},
        ...
        HumanLoopConfig=humanLoopConfig
    )
    return response

analyzeDocumentResponse = analyze_document_with_a2i(document, S3_BUCKET)
```

Add your human loop configuration to the Amazon Textract API call

Start Human Loop With Amazon Comprehend



Start Human Loop With Amazon Comprehend

```
reviews = ["I enjoy this product", "It is okay"]
CONFIDENCE_SCORE_THRESHOLD = 0.90

for review in reviews:
    # Call the a Amazon Comprehend Custom model
    response = comprehend.classify_document(
                                        Text=review,
                                        EndpointArn=comprehend_endpoint_arn)

    sentiment = response["Classes"][0]["Name"]
    confidence_score = response["Classes"][0]["Score"]
    print(f'Processing sample_review: "{review}"')
    ...
```

Sample reviews and confidence sore threshold

Get prediction response from a custom Amazon Comprehend model

Parse sentiment class and confidence score from model response

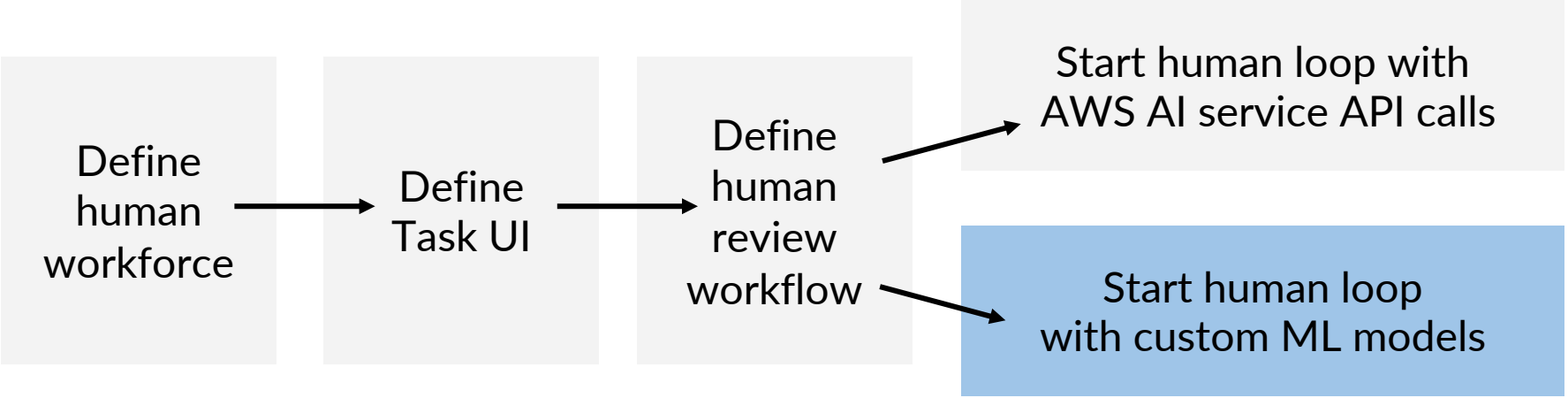
Start Human Loop With Amazon Comprehend

```
...
# Condition when to engage a human for review
if confidence_score < CONFIDENCE_SCORE_THRESHOLD:
    humanLoopName = <NAME>
    inputContent = {"initialValue": sentiment, "taskObject": sample_review}


    start_loop_response = a2i.start_human_loop(
        HumanLoopName=humanLoopName,
        FlowDefinitionArn=augmented_ai_flow_definition_arn,
        HumanLoopInput={"InputContent": json.dumps(inputContent)},
    )
```

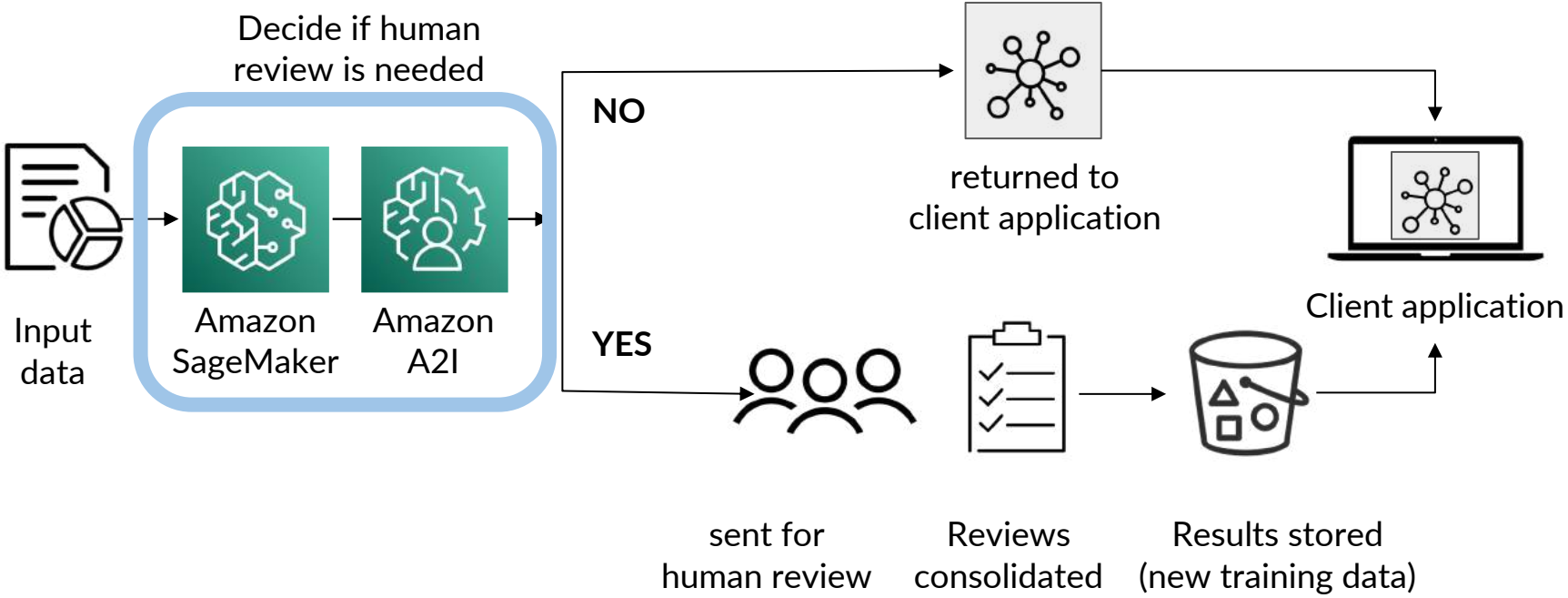
Check returned confidence score

Start Amazon A2I human loop with inputs

Steps


Define human workforce
Define Task UI
Define human review workflow
Start human loop with AWS AI service API calls
Start human loop with custom ML models

Use Amazon A2I With Amazon SageMaker



Use Amazon A2I With Amazon SageMaker

```
from sagemaker.predictor import Predictor
from sagemaker.serializers import JSONLinesSerializer
from sagemaker.deserializers import JSONLinesDeserializer

class SentimentPredictor(Predictor):
    def __init__(self, endpoint_name, sagemaker_session):
        super().__init__(endpoint_name,
                         sagemaker_session=...,
                         serializer=...,
                         deserializer=...)
```

Specify how to process model inputs (serializer) and model outputs (deserializer).

Use Amazon A2I With Amazon SageMaker

```
from sagemaker.pytorch.model import PyTorchModel
pytorch_model_name = 'model-{}'.format(timestamp)

model = PyTorchModel(name=...,
                                          predictor_cls=...,
                     model_data='s3://.../model.tar.gz',
                     ...)

pytorch_endpoint_name = 'endpoint-{}'.format(timestamp)

predictor = model.deploy(endpoint_name=pytorch_endpoint_name, ...)
```

Specify custom predictor

S3 location of model artifact

Deploy model endpoint

Use Amazon A2I With Amazon SageMaker

```
reviews = ["I enjoy this product", "It is okay"]

for review in reviews:
    inputs = [
        {"features": [review]},
    ]

    response = predictor.predict(inputs)

    prediction = response[0]['predicted_label']
    confidence_score = response[0]['probability']
    ...
```

Sample reviews

Call the model through the predictor.predict()

Parse predicted label and confidence score from model response

Use Amazon A2I With Amazon SageMaker

```
...
if confidence_score < CONFIDENCE_SCORE_THRESHOLD:

    human_loop_name = <NAME>
    input_content = {"initialValue": ..., "taskObject": ...}

    start_loop_response = a2i.start_human_loop(
        HumanLoopName=human_loop_name,
        FlowDefinitionArn=augmented_ai_flow_definition_arn,
        HumanLoopInput={"InputContent": json.dumps(input_content)},
    )
      ...
```

Check returned confidence score

Start Amazon A2I human loop with inputs

Human Loops Started

```
[{'probability': 0.937, 'predicted_label': 1}]
Checking prediction confidence 0.937 for sample review: "I enjoy this product"
Confidence score of 93.7% for star rating of 1 is above threshold of 90.0%
No human loop created.

[{'probability': 0.542, 'predicted_label': 1}]
Checking prediction confidence 0.542 for sample review: "It is okay"
Confidence score of 54.2% for prediction of 1 is less than the threshold of 90.0%
*** ==> Starting human loop with name: 1620962189-045527
```

Verify Results

input_content	human_answer
{'initialValue': 1, 'taskObject': 'It is okay'}	{'sentiment': {'label': '0'}}