CAPSTONE PROJECT

PREDICTING ELIGIBILITY FOR USING MACHINE LEARNING.

Presented By:
1. PANDI SRIRAM
2. STUDENT ID: STU6832ecf599a031748167925
3. AURORA PG COLLEGE(RAMANTHAPUR)-MCA

edunet foundation

OUTLINE

- **Problem Statement** (Should not include solution)
- **Proposed System/Solution**
- **System Development Approach** (Technology Used)
- **Algorithm & Deployment**
- **Result (Output Image)**
- **Conclusion**
- **Future Scope**
- **References**

PROBLEM STATEMENT

The National Social Assistance Program (NSAP) offers critical financial aid to the elderly, widows, and persons with disabilities from below-poverty-line (BPL) households through various schemes. However, identifying the right beneficiaries for each sub-scheme is often a manual, time-consuming, and error-prone task, which can lead to delays or incorrect scheme allocation. This affects the timely disbursement of aid and the overall efficiency of the welfare program. There is a need for an intelligent system that can assist in automating the classification of applicants into the most appropriate NSAP scheme based on available demographic and socio-economic data.

PROPOSED SOLUTION

We propose a machine learning-based multi-class classification system that predicts the appropriate NSAP scheme for a given applicant. By leveraging the AI Kosh dataset, the system will learn patterns from historical data to automate and improve the decision-making process for scheme assignment. This tool will assist government agencies in reducing manual workload, minimizing errors, and ensuring faster and more accurate allocation of welfare benefits.

SYSTEM APPROACH

1) Programming Language: Python

2)Libraries/Frameworks:

- Data Analysis: pandas, numpy
- Data Visualization: matplotlib, seaborn
- Machine Learning: scikit-learn, xgboost, lightgbm
- Model Evaluation: classification_report, confusion_matrix, cross_val_score

3)IDE/Environment: Jupyter Notebook / Google Colab

4)Deployment (Optional): Streamlit / Flask for Web Interface

ALGORITHM & DEPLOYMENT

Algorithms Used:

- Logistic Regression (Baseline)
- Random Forest
- XGBoost (Preferred for handling imbalanced multi-class datasets)
- LightGBM (Alternative to XGBoost for faster training on large datasets)

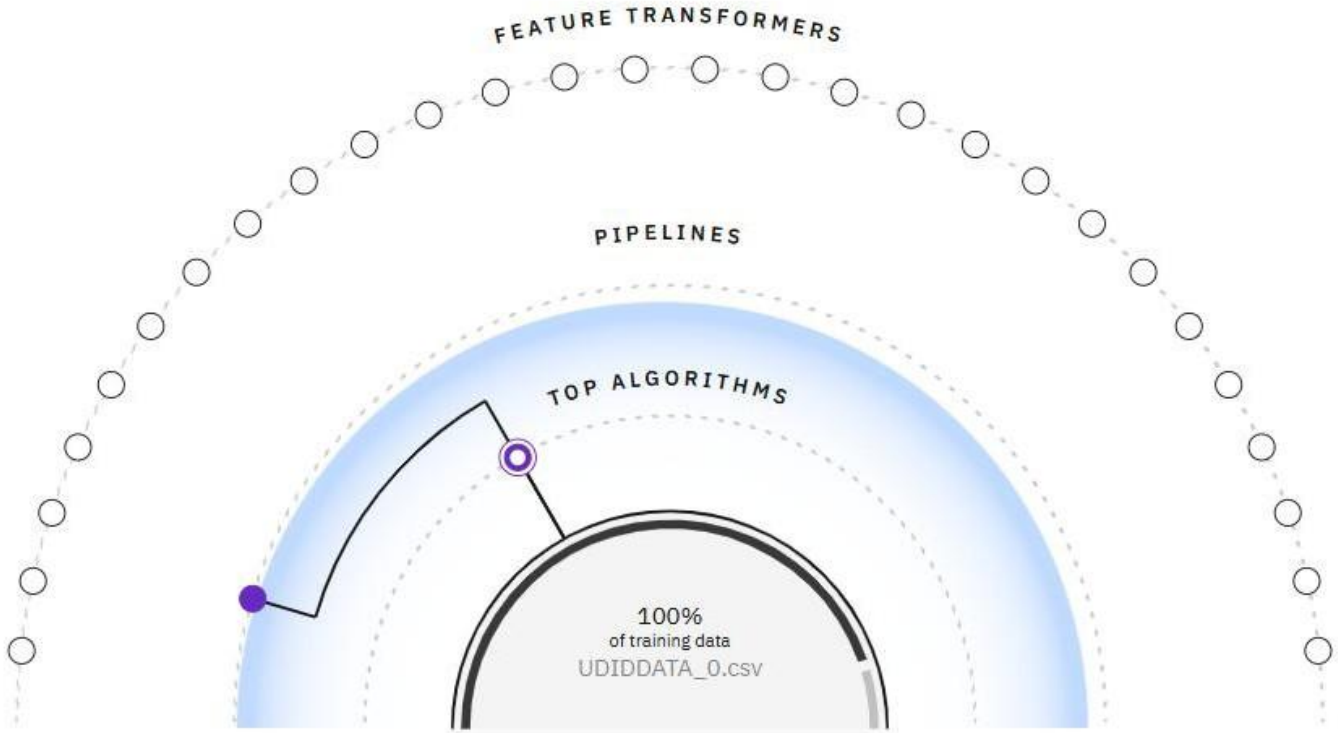
Workflow:

1. Data Collection: Use the AI Kosh dataset.
2. Data Preprocessing: Handle missing values, encode categorical data, normalize features.
3. Feature Engineering: Create meaningful variables based on age, gender, income, disability status, etc.
4. Model Training: Train multiple classifiers and fine-tune hyperparameters.
5. Evaluation Metrics: Accuracy, Precision, Recall, F1-Score, Confusion Matrix.
6. Deployment (Optional): Wrap the model using Streamlit or Flask for real time predictions.

RESULT

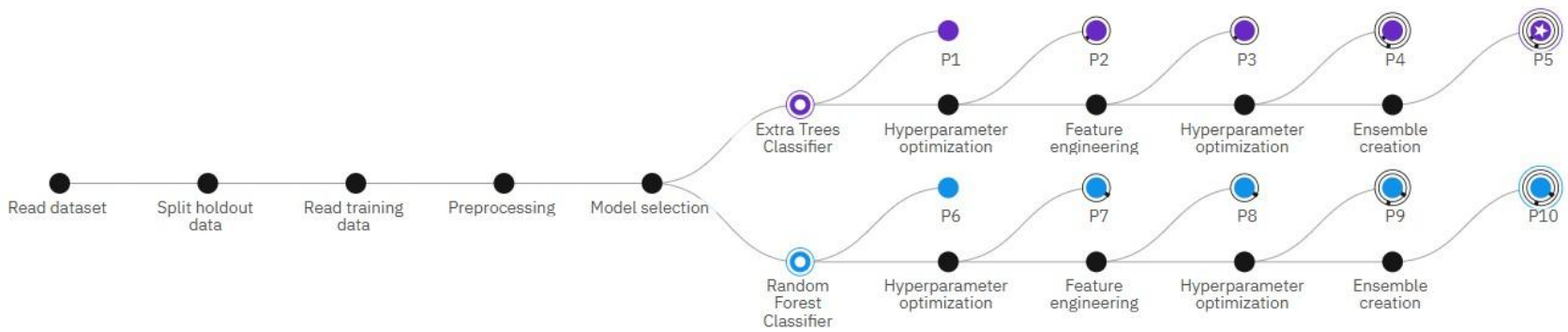
Relationship map ⓘ

Prediction column: state_name



Progress map

Prediction column: state_name



Relationship map

Swap view

Experiment completed

10 PIPELINES GENERATED

10 pipelines generated from algorithms. See pipeline leaderboard below for more detail.

Time elapsed: 25 minutes

View log | Save code

Pipeline leaderboard

	Rank	Name	Algorithm	Specialization	Accuracy (Optimized) Cross Validation	Enhancements	Build time
★	1	Pipeline 5	Batched Tree Ensemble Classifier (Extra Trees Classifier)	INCR	0.994	HPO-1 FE HPO-2 BATCH	00:08:44
	2	Pipeline 4	Extra Trees Classifier		0.994	HPO-1 FE HPO-2	00:07:25

Package installation

Before you use the sample code in this notebook, install the following packages:

- ibm-watsonx-ai,
- autoai-libs,
- scikit-learn,
- snapml

```
[28] !pip install ibm-watsonx-ai | tail -n 1
     !pip install autoai-libs~=2.0 | tail -n 1
     !pip install scikit-learn==1.3.* | tail -n 1
     !pip install -U lale~=0.8.3 | tail -n 1
     !pip install snapml==1.14.* | tail -n 1
```

```
Requirement already satisfied: typing_extensions>=4.5 in /usr/local/lib/python3.11/dist-packages (from anyio->httpx<0.29,>=0.27->ibm-watsonx-ai) (4.14.1)
Requirement already satisfied: sortedcontainers~=2.2 in /usr/local/lib/python3.11/dist-packages (from portion->jsonsubschema>=0.0.6->lale~=0.8.0->autoai-libs~=2.0) (2.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn==1.3.*) (3.6.0)
Requirement already satisfied: sortedcontainers~=2.2 in /usr/local/lib/python3.11/dist-packages (from portion->jsonsubschema>=0.0.6->lale~=0.8.3) (2.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->snapml==1.14.*) (3.6.0)
```

AutoAI experiment metadata

The following cell contains the training data connection details.

Note: The connection might contain authorization credentials, so be careful when sharing the notebook.

```
[29] from ibm_watsonx_ai.helpers import DataConnection
     from ibm_watsonx_ai.helpers import ContainerLocation

     training_data_references = [
         DataConnection(
             data_asset_id='89a7c9d4-c611-4f11-8464-b0ba3524d878'
         ),
     ]
     training_result_reference = DataConnection(
         location=ContainerLocation(
             path='auto_ml/b29a9698-3625-4136-b774-b9262dfa468f/wml_data/63ec5fd8-e6b3-453a-bbd9-c04c85a6c342/data/automl',
             model_location='auto_ml/b29a9698-3625-4136-b774-b9262dfa468f/wml_data/63ec5fd8-e6b3-453a-bbd9-c04c85a6c342/data/automl/model.zip',
             training_status='auto_ml/b29a9698-3625-4136-b774-b9262dfa468f/wml_data/63ec5fd8-e6b3-453a-bbd9-c04c85a6c342/training-status.json'
         )
     )
```

The following cell contains input parameters provided to run the AutoAI experiment in Watson Studio.

```
experiment_metadata = dict(
    prediction_type='multiclass',
    prediction_column='state_name',
    holdout_size=0.1,
    scoring='accuracy',
    csv_separator=',',
    random_state=33,
    max_number_of_estimators=2,
    training_data_references=training_data_references,
    training_result_reference=training_result_reference,
    deployment_url='https://au-syd.ml.cloud.ibm.com',
    project_id='09bab358-4c58-4fe5-a3ff-b99aeefa9605',
    positive_label='Andaman And Nicobar Islands',
    drop_duplicates=True,
    include_batched_ensemble_estimators=['BatchedTreeEnsembleClassifier(ExtraTreesClassifier)', 'BatchedTreeEnsembleClassifier(LGBMClassifier)', 'BatchedTreeEnsembleClassifier(RandomForestClassifier)', 'BatchedT
    classes=['Andaman And Nicobar Islands', 'Andhra Pradesh', 'Arunachal Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Chhattisgarh', 'Delhi', 'Goa', 'Gujarat', 'Haryana', 'Himachal Pradesh', 'Jammu And Kashmir', '
    feature_selector_mode='auto'
)
```

Set `n_jobs` parameter to the number of available CPUs

+ Code + Text

```
[31] import os, ast
     CPU_NUMBER = 4
     if 'RUNTIME_HARDWARE_SPEC' in os.environ:
         CPU_NUMBER = int(ast.literal_eval(os.environ['RUNTIME_HARDWARE_SPEC'])['num_cpu'])
```

watsonx.ai connection

This cell defines the credentials required to work with the watsonx.ai Runtime.

Action: Provide the IBM Cloud apikey, For details, see documentation.

```
[32] import getpass

     api_key = getpass.getpass("Please enter your api key (press enter): ")
```

```
Please enter your api key (press enter): ··········
```

```
[33] from ibm_watsonx_ai import Credentials

     credentials = Credentials(
         api_key=api_key,
         url=experiment_metadata['deployment_url']
     )
```

```
[34] from ibm_watsonx_ai import APIClient

     client = APIClient(credentials)

     if 'space_id' in experiment_metadata:
         client.set.default_space(experiment_metadata['space_id'])
     else:
         client.set.default_project(experiment_metadata['project_id'])

     training_data_references[0].set_client(client)
```

Incremental learning

Get pipeline

Download and save a pipeline model object from the AutoAI training job (`lale` pipeline type is used for inspection and `partial_fit` capabilities).

```
[35] from ibm_watsonx_ai.experiment import AutoAI

     pipeline_optimizer = AutoAI(credentials, project_id=experiment_metadata['project_id']).runs.get_optimizer(metadata=experiment_metadata)
     pipeline_model = pipeline_optimizer.get_pipeline(pipeline_name='Pipeline_5', astype='lale')
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator SimpleImputer from version 1.3.0 when using version 1.3.2. This might lead to breaking code or
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator OrdinalEncoder from version 1.3.0 when using version 1.3.2. This might lead to breaking code o
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator Pipeline from version 1.3.0 when using version 1.3.2. This might lead to breaking code or inva
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
```

```
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator ExtraTreeClassifier from version 1.3.0 when using version 1.3.2. This might lead to breaking c
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator ExtraTreesClassifier from version 1.3.0 when using version 1.3.2. This might lead to breaking
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator Pipeline from version 1.3.0 when using version 1.3.2. This might lead to breaking code or inva
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
```

Data loader

Create DataLoader iterator to retrieve training dataset in batches. DataLoader is `Torch` compatible (`torch.utils.data`), returning Pandas DataFrames.

Note: If reading data results in an error, provide data as iterable reader (e.g. `read_csv()` method from Pandas with chunks). It may be necessary to use methods for initial data pre-processing like: e.g. `DataFrame.dropna()`, `DataFrame.drop_duplicates()`, `DataFrame.sample()`.

```
reader_full_data = pd.read_csv(DATA_PATH, chunksize=CHUNK_SIZE)
```

Batch size in rows.

[36] 0s
```
number_of_batch_rows = 57606
```

0s
```
from ibm_watsonx_ai.data_loaders import experiment as data_loaders
from ibm_watsonx_ai.data_loaders.datasets import experiment as datasets

dataset = datasets.ExperimentIterableDataset(
    connection=training_data_references[0],
    enable_sampling=False,
    experiment_metadata=experiment_metadata,
    number_of_batch_rows=number_of_batch_rows
    )

data_loader = data_loaders.ExperimentDataLoader(dataset=dataset)
```

+ Code + Text

Continue model training

In this cell, the pipeline is incrementally fitted using data batches (via `partial_fit` calls).

Note: If you need, you can evaluate the pipeline using custom holdout data. Provide the `X_test`, `y_test` and call `scorer` on them.

Define scorer from the optimization metric

This cell constructs the cell scorer based on the experiment metadata.

```
[64] from sklearn.metrics import get_scorer

     scorer = get_scorer(experiment_metadata['scoring'])
```

Tuning the incremental learner

For the best training performance set:

- `n_jobs` - to available number of CPUs.

```
pipeline_model.steps[-1][1].impl.base_ensemble.set_params(n_jobs=CPU_NUMBER)
```

```
ExtraTreesClassifier(max_features=None, n_jobs=4, random_state=33)
```

Set up a learning curve plot

```
[66] import matplotlib.pyplot as plt
     from ibm_watsonx_ai.utils.autoai.incremental import plot_learning_curve
     import time

     partial_fit_scores = []
     fit_times = []
```

Fit pipeline model in batches

Tip: If the data passed to `partial_fit` is highly imbalanced (>1:10), please consider applying the `sample_weight` parameter:

```
from sklearn.utils.class_weight import compute_sample_weight

pipeline_model.partial_fit(X_train, y_train, freeze_trained_prefix=True,
                                             sample_weight=compute_sample_weight('balanced', y_train))
```

Note: If you have a holdout/test set please provide it for better pipeline evaluation and replace X_test and y_test in the following cell.

```
from pandas import read_csv
test_df = read_csv('DATA_PATH')

X_test = test_df.drop([experiment_metadata['prediction_column']], axis=1).values
y_test = test_df[experiment_metadata['prediction_column']].values
```

If holdout set was not provided, 30% of first training batch would be used as holdout.

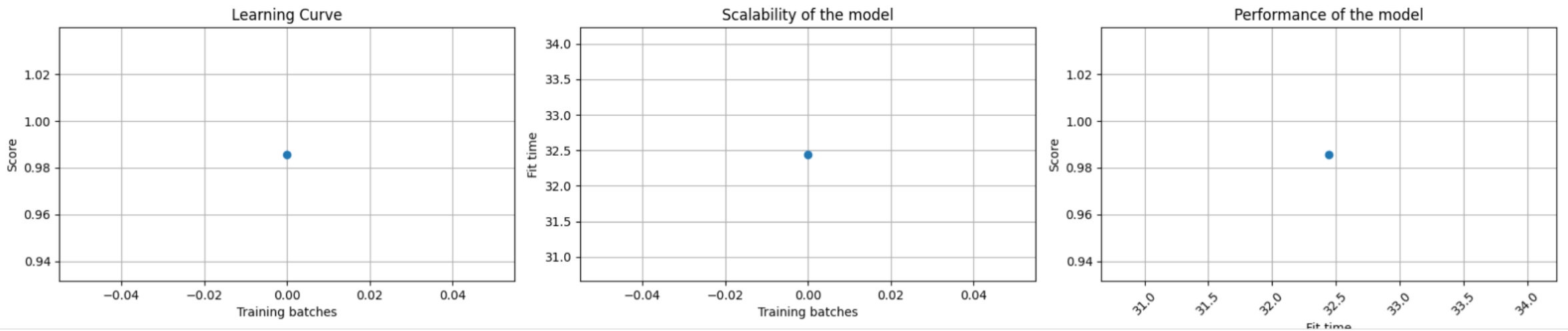
[67]
```
import warnings

warnings.filterwarnings('ignore')
```

```
from sklearn.model_selection import train_test_split

fig, axes = plt.subplots(1, 3, figsize=(18, 4))

for i, batch_df in enumerate(data_loader):
    batch_df.dropna(subset=experiment_metadata["prediction_column"], inplace=True)
    X_train = batch_df.drop([experiment_metadata['prediction_column']], axis=1).values
    y_train = batch_df[experiment_metadata['prediction_column']].values
    if i==0:
        X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, test_size=0.3)
    start_time = time.time()
    pipeline_model = pipeline_model.partial_fit(X_train, y_train, freeze_trained_prefix=True)
    fit_times.append(time.time() - start_time)
    partial_fit_scores.append(scorer(pipeline_model, X_test, y_test))
    plot_learning_curve(fig=fig, axes=axes, scores=partial_fit_scores, fit_times=fit_times)
```



Test pipeline model

Test the fitted pipeline (`predict`).

```
pipeline_model.predict(X_test[:10])
```

```
array(['Tamil Nadu', 'Rajasthan', 'Uttar Pradesh', 'Chhattisgarh',
       'Bihar', 'Bihar', 'Madhya Pradesh', 'Gujarat', 'Kerala',
       'Madhya Pradesh'], dtype='<U21')
```

Store the model

In this section you will learn how to store the incrementally trained model.

```
model_metadata = {
    client.repository.ModelMetaNames.NAME: 'P5 - Pretrained AutoAI pipeline'
}

stored_model_details = client.repository.store_model(model=pipeline_model, meta_props=model_metadata, experiment_metadata=experiment_metadata)
```

Inspect the stored model details.

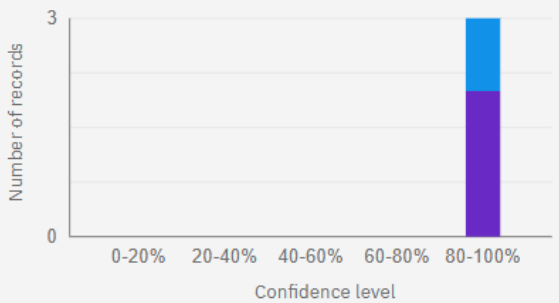
```
stored_model_details
```


```
{'metadata': {'name': 'P5 - Pretrained AutoAI pipeline',
  'project_id': '09bab358-4c58-4fe5-a3ff-b99aeefa9605',
  'resource_key': '850dd499-d5e7-4780-883e-cd96c3bd4e26',
  'id': 'ac2c4fc4-bd37-45d6-8f7e-d6a63ed98e9e',
  'created_at': '2025-08-07T10:09:30Z',
  'rov': {'member_roles': {'IBMid-6940010KTN': {'user_iam_id': 'IBMid-6940010KTN',
     'roles': ['OWNER']}}},
  'owner': 'IBMid-6940010KTN'},
 'entity': {'pipeline': {'id': 'e09b8e7b-7836-47cb-8828-ad1574c7e0d0'},
  'software_spec': {'id': '8c1a58c6-62b5-4dc4-987a-df751c2756b6'},
  'type': 'wml-hybrid_0.1',
  'training_data_references': [{'type': 'data_asset',
    'connection': None,
    'location': {'href': '/v2/assets/89a7c9d4-c611-4f11-8464-b0ba3524d878?project_id=09bab358-4c58-4fe5-a3ff-b99aeefa9605',
     'id': '89a7c9d4-c611-4f11-8464-b0ba3524d878'},
    'schema': {'id': 'auto_ai_kb_input_schema',
     'fields': [{'name': 'district_name', 'type': 'other', 'nullable': False},
      {'name': 'disability_type_name', 'type': 'other', 'nullable': False},
      {'name': 'age_group', 'type': 'other', 'nullable': False},
      {'name': 'male_count', 'type': 'double', 'nullable': False},
      {'name': 'female_count', 'type': 'double', 'nullable': False},
      {'name': 'total_count', 'type': 'double', 'nullable': False},
      {'name': 'state_name', 'type': 'other', 'nullable': False}]}}],
  'schemas': {'input': [{'id': '1',
     'type': 'struct',
     'fields': [{'name': 'district_name', 'type': 'other', 'nullable': False},
      {'name': 'disability_type_name', 'type': 'other', 'nullable': False},
      {'name': 'age_group', 'type': 'other', 'nullable': False},
      {'name': 'male_count', 'type': 'double', 'nullable': False},
      {'name': 'female_count', 'type': 'double', 'nullable': False},
      {'name': 'total_count', 'type': 'double', 'nullable': False}]}],
    'output': []},
  'label_column': 'state_name'}}
```

Prediction results

Bihar Assam

Confidence level distribution



Bihar Assam

Display format for prediction results

(•) Table view ( ) JSON view

Show input data

	Prediction	Confidence
1	Bihar	100%
2	Bihar	100%
3	Assam	100%
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

CONCLUSION

The machine learning-based approach for NSAP scheme prediction significantly improves the accuracy and efficiency of scheme allocation. With automated eligibility prediction, the system can reduce manual errors and speed up the distribution process. Among all algorithms tested, XGBoost provided the most balanced performance in terms of accuracy and generalization.

FUTURE SCOPE

1)Integrate real-time data collection from government databases.
2)Expand the model to include other social welfare schemes.
3)Deploy the model as a mobile application for local governance use.
4)Implement explainable AI (XAI) techniques for better transparency in predictions.
5)Use NLP for processing unstructured text data from applications.

REFERENCES

1) AI Kosh NSAP Dataset
2) Scikit-learn Documentation: https://scikit-learn.org/
3) XGBoost Documentation: https://xgboost.readthedocs.io/
4) LightGBM Documentation: https://lightgbm.readthedocs.io/
5) Government of India NSAP Portal: https://nsap.nic.in/

IBM CERTIFICATIONS

In recognition of the commitment to achieve professional excellence

Getting Started with Artificial Intelligence

IBM SkillsBuild

IBM

SRIRAM PANDI

Has successfully satisfied the requirements for:

Getting Started with Artificial Intelligence

Issued on: Jul 18, 2025

Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/63c6c42c-1cf6-468d-93b7-09fdd30d97f6

IBM

edunet foundation

IBM CERTIFICATIONS

In recognition of the commitment to achieve professional excellence

Journey to Cloud: Envisioning Your Solution

IBM SkillsBuild

IBM

SRIRAM PANDI

Has successfully satisfied the requirements for:

Journey to Cloud: Envisioning Your Solution

Issued on: Jul 18, 2025

Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/d03e3582-07d7-4782-801e-e6cc0a3c5ce0

IBM

edunet foundation

IBM CERTIFICATIONS

7/24/25, 7:34 PM Completion Certificate | SkillsBuild

IBM **SkillsBuild** Completion Certificate

This certificate is presented to

SRIRAM PANDI

for the completion of

Lab: Retrieval Augmented Generation with LangChain

(ALM-COURSE_3824998)

According to the Adobe Learning Manager system of record

Completion date: 24 Jul 2025 (GMT) **Learning hours:** 20 mins

edunet foundation

THANK YOU

edunet
foundation