CAPSTONE PROJECT

PREDICTING ELIGIBILITY FOR USING MACHINE LEARNING.

Presented By:

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

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

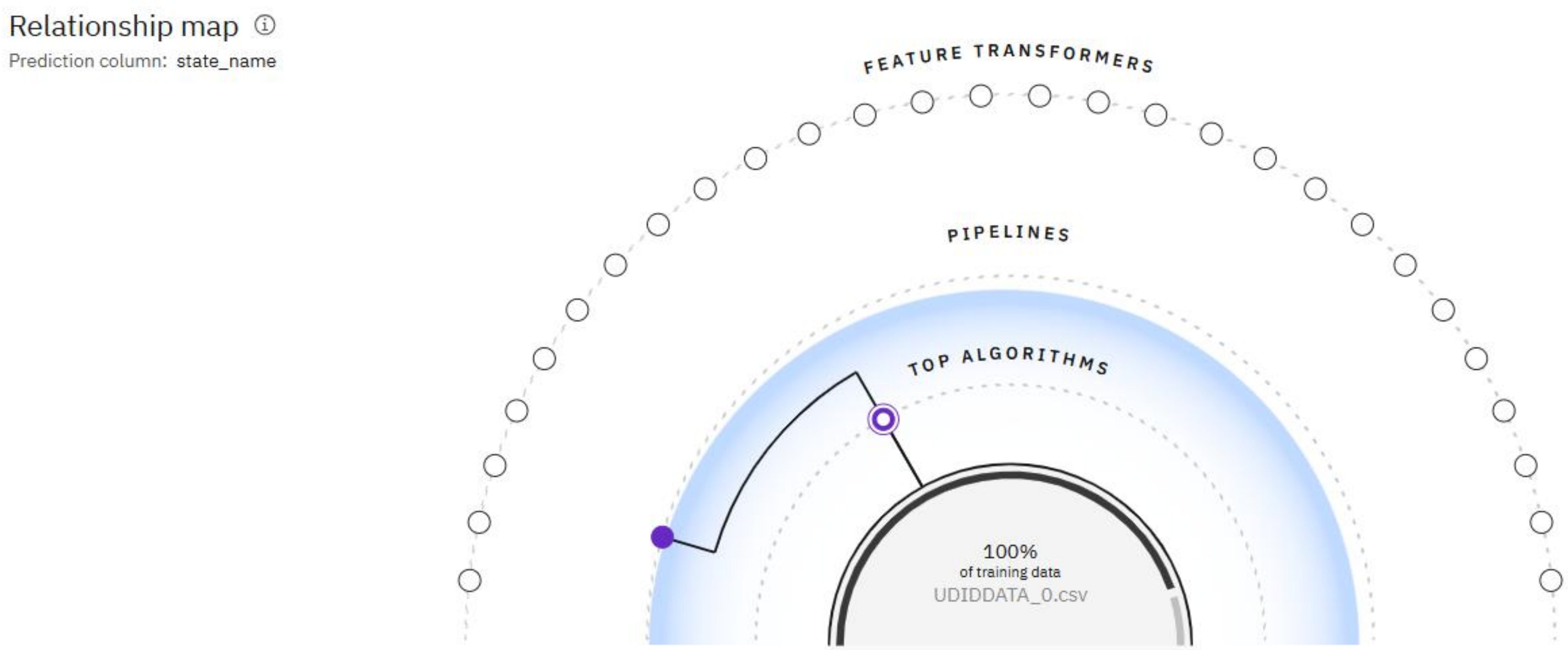
Algorithms Used:

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

Workflow:

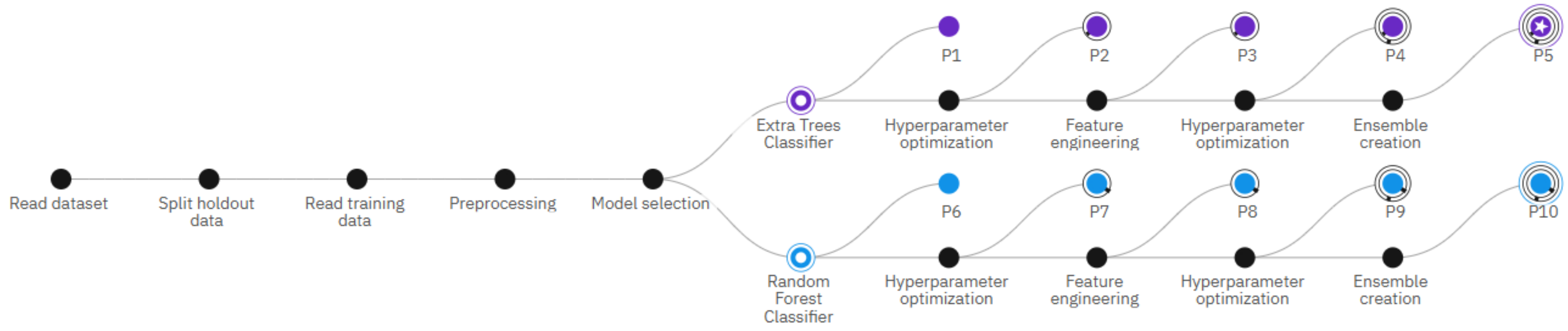
1. **Data Collection:** Use the AI Kosh dataset.
2. **Data Preprocessing:** Handle missing values, encode categorical data, normalize features.
3. **Feature Engineering:** Create meaningful variables based on age, gender, income, disability status, etc.
4. **Model Training:** Train multiple classifiers 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
FEATURE TRANSFORMERS
PIPELINES
TOP ALGORITHMS
100%
of training data
UDIDDATA_0.csv

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,
- lale,
- scikit-learn,
- xgboost,
- lightgbm,
- snapml

```
!pip install ibm-watsonx-ai | tail -n 1
!pip install autoai-libs~=2.0 | tail -n 1
!pip install -U 'lale~=0.8.3' | tail -n 1
!pip install scikit-learn==1.3.* | tail -n 1
!pip install xgboost==2.0.* | tail -n 1
!pip install lightgbm==4.2.* | tail -n 1
!pip install snapml==1.14.* | tail -n 1
```

Experiment configuration

Create a job

Experiment metadata

This cell defines the metadata for the experiment, including: training_data_references, training_result_reference, experiment_metadata.

```
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'
    )
)
```

```
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)', 'BatchedTreeEnsembleCla
    feature_selector_mode='auto'
)
```

edunet foundation

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.

```
import getpass

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

```
from ibm_watsonx_ai import Credentials

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

Create a job

Get fitted AutoAI optimizer

```
from ibm_watsonx_ai.experiment import AutoAI

pipeline_optimizer = AutoAI(credentials, project_id=experiment_metadata['project_id']).runs.get_optimizer(metadata=experiment_metadata)
```

Use `get_params()` to retrieve configuration parameters.

```
pipeline_optimizer.get_params()
```

Pipelines comparison

Use the `summary()` method to list trained pipelines and evaluation metrics information in the form of a Pandas DataFrame. You can use the DataFrame to compare all discovered pipelines and select the one you like for further testing.

```
summary = pipeline_optimizer.summary()
best_pipeline_name = list(summary.index)[0]
summary
```

Get pipeline as a scikit-learn pipeline model

After you compare the pipelines, download and save a scikit-learn pipeline model object from the AutoAI training job.

Tip: To get a specific pipeline, pass the pipeline name in:

```
pipeline_optimizer.get_pipeline(pipeline_name=pipeline_name)
```

```
pipeline_model = pipeline_optimizer.get_pipeline()
```

Next, check the importance of features for selected pipeline.

```
pipeline_optimizer.get_pipeline_details()['features_importance']
```

Tip: If you want to check all the details of the model evaluation metrics, use:

```
pipeline_optimizer.get_pipeline_details()
```

Score the fitted pipeline with the generated scorer using the holdout dataset.

1. Get sklearn pipeline_model

```
sklearn_pipeline_model = pipeline_optimizer.get_pipeline(astype=AutoAI.PipelineTypes.SKLEARN)
```

2. Get training and testing data

```
from ibm_watsonx_ai import APIClient

client = APIClient(credentials=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)
```

```
_, X_test, _, y_test = training_data_references[0].read(experiment_metadata=experiment_metadata, with_holdout_split=True, use_flight=True)
```

3. Define scorer, score the fitted pipeline with the generated scorer using the holdout dataset.

```
from sklearn.metrics import get_scorer

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

```
score = scorer(sklearn_pipeline_model, X_test.values, y_test.values)
print(score)
```

Deployment creation

```
target_space_id = input("Enter your space ID here (press enter): ")
```

```
from ibm_watsonx_ai.deployment import WebService

service = WebService(
    source_instance_credentials=credentials,
    target_instance_credentials=credentials,
    source_project_id=experiment_metadata['project_id'],
    target_space_id=target_space_id
)
service.create(
    model=best_pipeline_name,
    metadata=experiment_metadata,
    deployment_name='Best_pipeline_webservice'
)
```

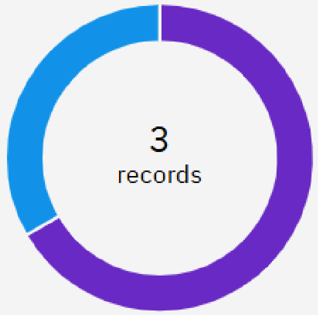
Use the `print` method for the deployment object to show basic information about the service:

```
print(service)
```

To show all available information about the deployment, use the `.get_params()` method.

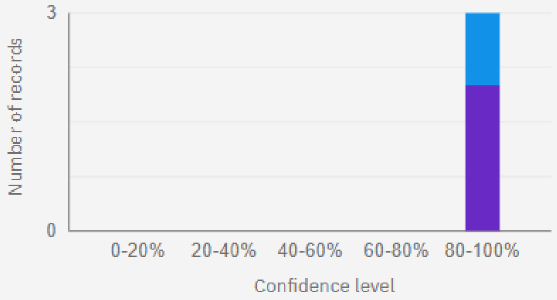
```
service.get_params()
```

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

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

IBM CERTIFICATIONS

In recognition of the commitment to achieve professional excellence

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

IBM CERTIFICATIONS

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