Nompumelelo Ngwenya

Deploying a machine learning model on Azure AI Platform.

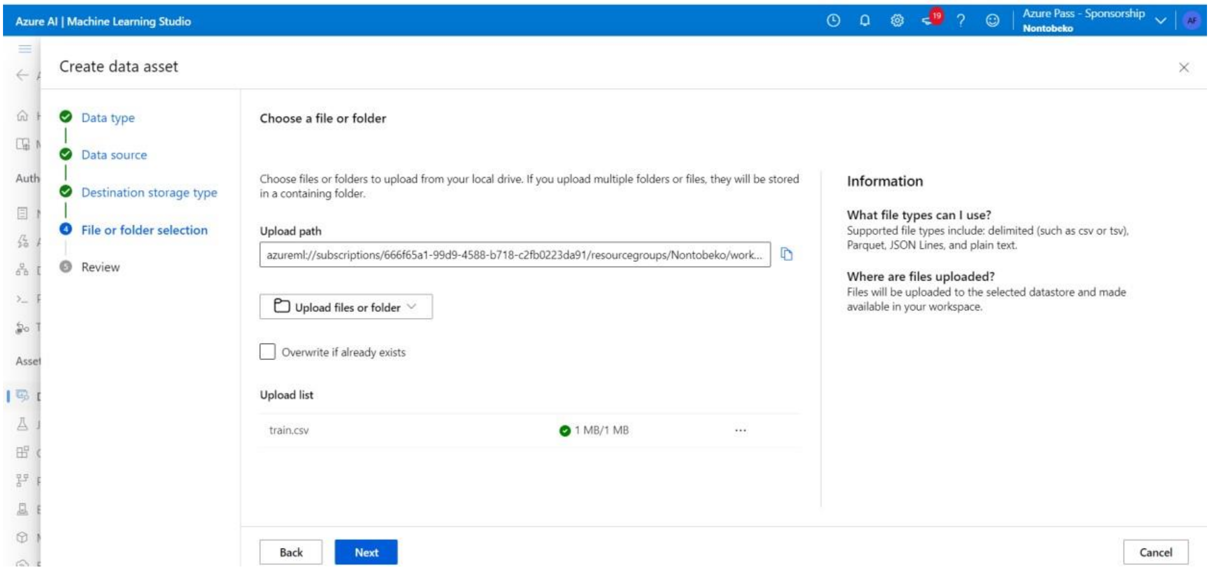
Introduction and Overview

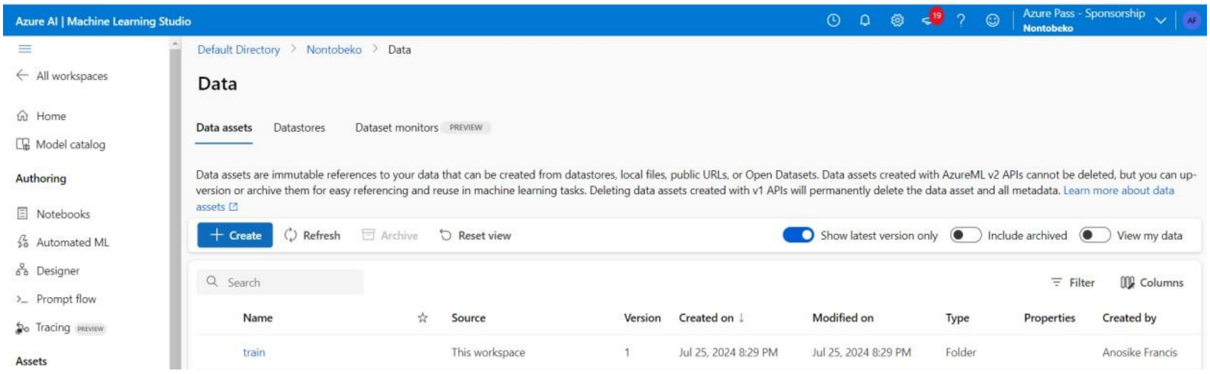
Purpose: This document aims to guide stakeholders through the process of deploying a machine learning model trained on the Employees Train dataset using Azure Machine Learning.

Audience: Data scientists, developers, and operations teams involved in model deployment and maintenance.

Data Source: (https://www.kaggle.com/datasets/stealthtechnologies/employee-attritiondataset?select=train.csv)

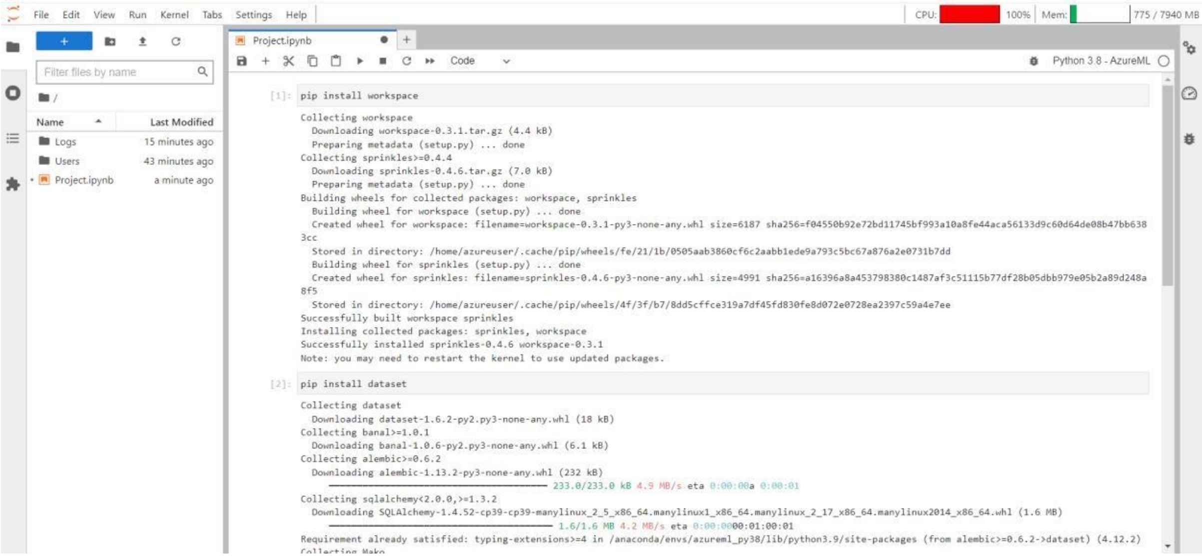
Upload the data into the Azure workspace

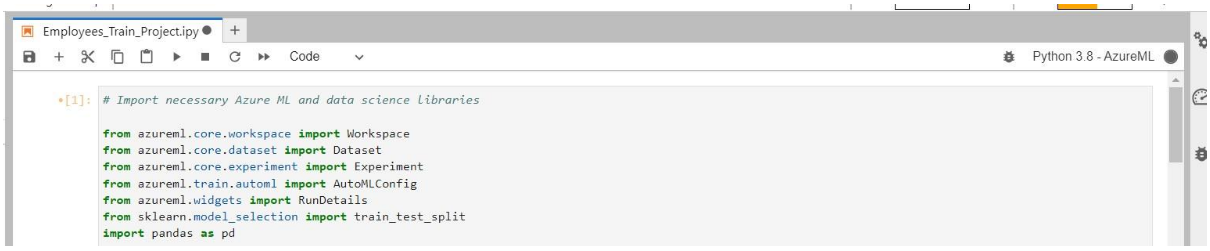




Create the compute instance and open the Jupyter Lab

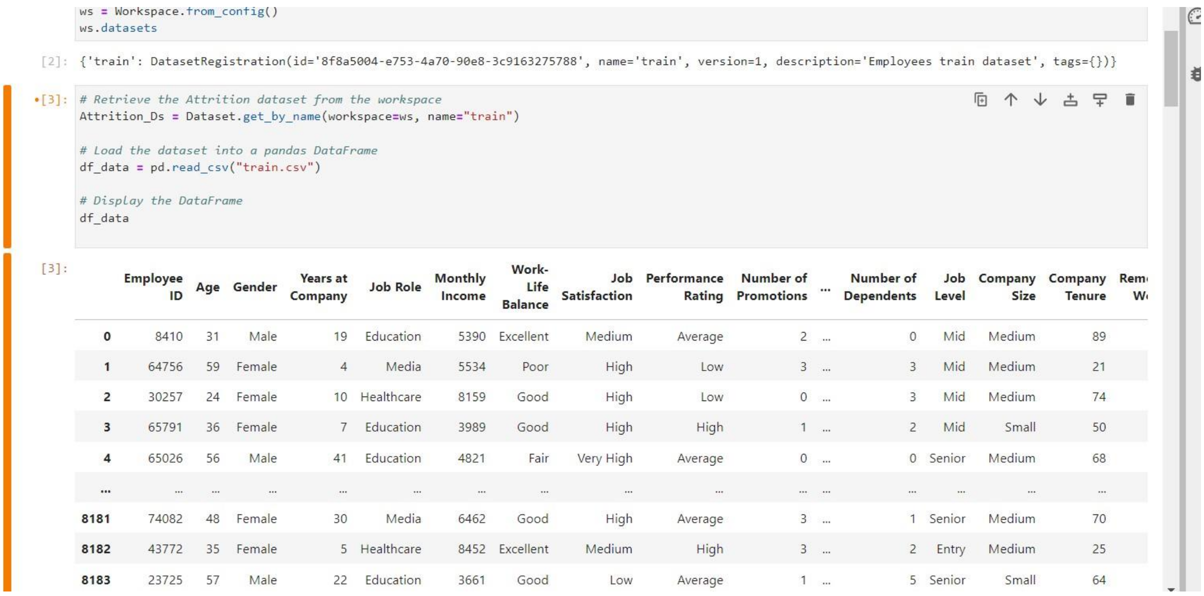
Import the necessary packages and libraries





Connecting to your workspace, This will allow you to access the data you uploaded on the azure workspace

Displaying the data on jupitor notebook



	Employee ID	Age	Gender	Years at Company	Job Role	Monthly Income	Work-Life Balance	Job Satisfaction	Performance Rating	Number of Promotions	...	Number of Dependents	Job Level	Company Size	Company Tenure	Rem W
0	8410	31	Male	19	Education	5390	Excellent	Medium	Average	2	...	0	Mid	Medium	89	
1	64756	59	Female	4	Media	5534	Poor	High	Low	3	...	3	Mid	Medium	21	
2	30257	24	Female	10	Healthcare	8159	Good	High	Low	0	...	3	Mid	Medium	74	
3	65791	36	Female	7	Education	3989	Good	High	High	1	...	2	Mid	Small	50	
4	65026	56	Male	41	Education	4821	Fair	Very High	Average	0	...	0	Senior	Medium	68	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8181	74082	48	Female	30	Media	6462	Good	High	Average	3	...	1	Senior	Medium	70	
8182	43772	35	Female	5	Healthcare	8452	Excellent	Medium	High	3	...	2	Entry	Medium	25	
8183	23725	57	Male	22	Education	3661	Good	Low	Average	1	...	5	Senior	Small	64	

Data Preprocessing**:**

This step involves preparing the dataset for model training, including handling missing values, scaling features if necessary, and splitting the dataset into training and testing sets.

```
[4]: # Display the columns of the DataFrame
     df_data.columns

[4]: Index(['Employee ID', 'Age', 'Gender', 'Years at Company', 'Job Role',
            'Monthly Income', 'Work-Life Balance', 'Job Satisfaction',
            'Performance Rating', 'Number of Promotions', 'Overtime',
            'Distance from Home', 'Education Level', 'Marital Status',
            'Number of Dependents', 'Job Level', 'Company Size', 'Company Tenure',
            'Remote Work', 'Leadership Opportunities', 'Innovation Opportunities',
            'Company Reputation', 'Employee Recognition', 'Attrition'],
           dtype='object')

[5]: # Split the data into training and testing sets with 70% for training and 30% for testing
     x_train, x_test = train_test_split(df_data, test_size = 0.3)

[6]: # Display the shape of the training set
     x_train.shape

[6]: (5730, 24)

[7]: # Display the shape of the testing set
     x_test.shape

[7]: (2456, 24)
```

CONFIGURATION SETTINGS

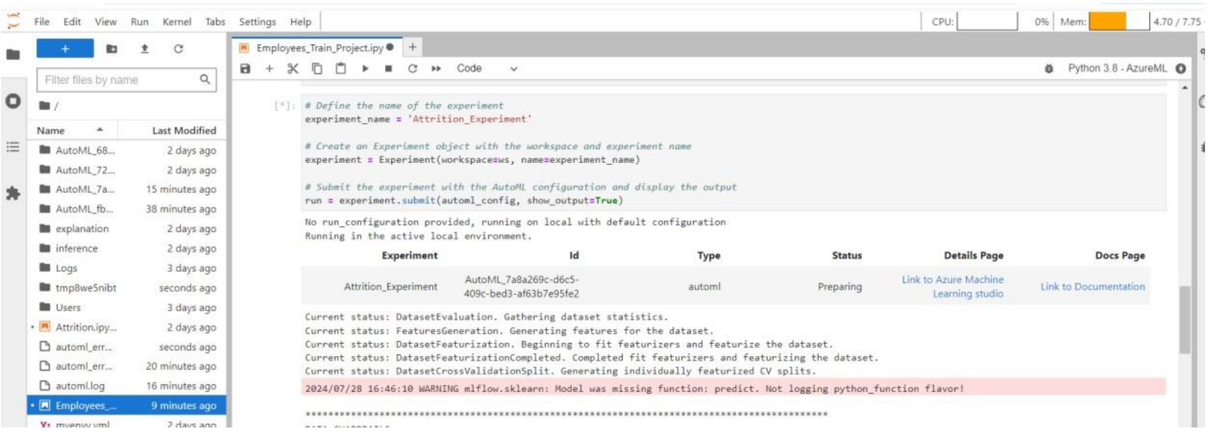
```
automl_settings = {
    "iteration_timeout_minutes": 2,
    "experiment_timeout_minutes": 15,
    "enable_early_stopping": True,
    "primary_metric": 'AUC_weighted',
    "featurization": 'auto',
    "n_cross_validations": 5
}

automl_config = AutoMLConfig(
    task='classification',                    debug_log='automl_errors.log',
training_data=x_train,                    label_column_name="Attrition",
    **automl_settings
)
```

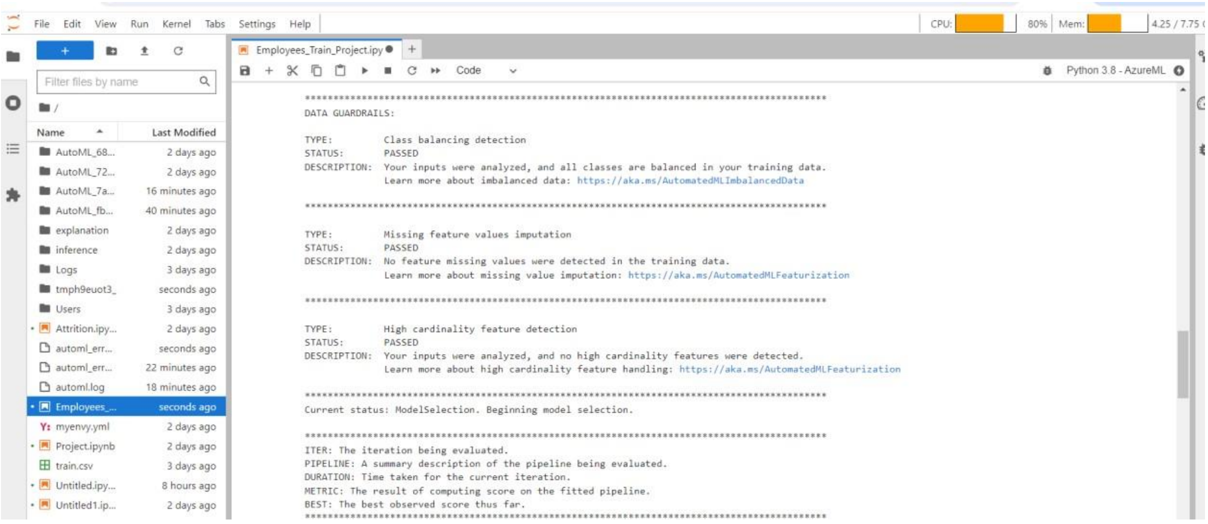
```
[11]: # Define the settings for the AutoML configuration

automl_settings = {
    "iteration_timeout_minutes": 2,
    "experiment_timeout_minutes": 15,
    "enable_early_stopping": True,
    "primary_metric": 'AUC_weighted',
    "featurization": 'auto',
    "n_cross_validations": 5
}
```

Configuring the settings

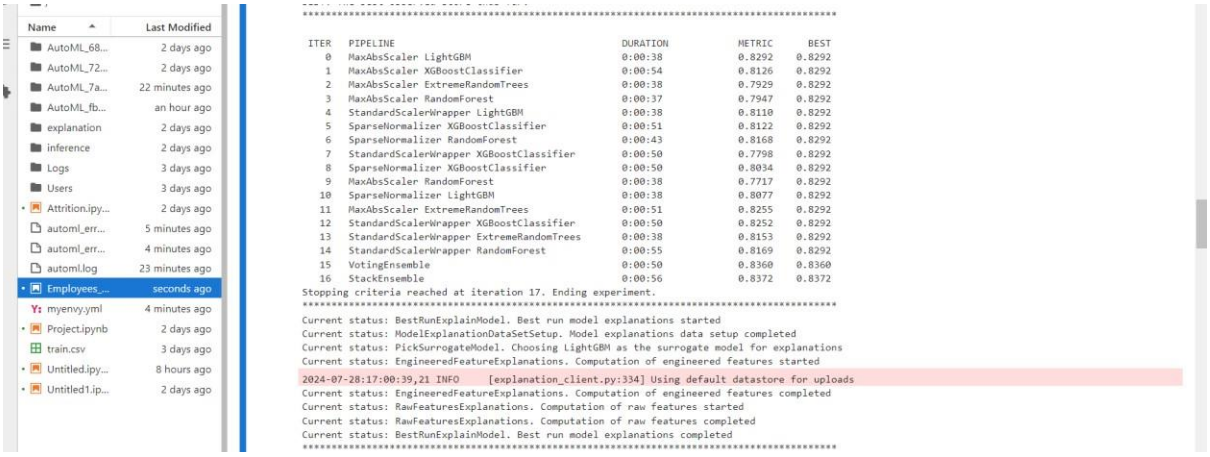
```
[12]: # Configure the AutoML settings for a classification task

automl_config = AutoMLConfig(
    task='classification',
    debug_log='automl_errors.log',
    training_data=x_train,
    label_column_name="Attrition",
    **automl_settings
)
```

Creating your experiment for the deployment



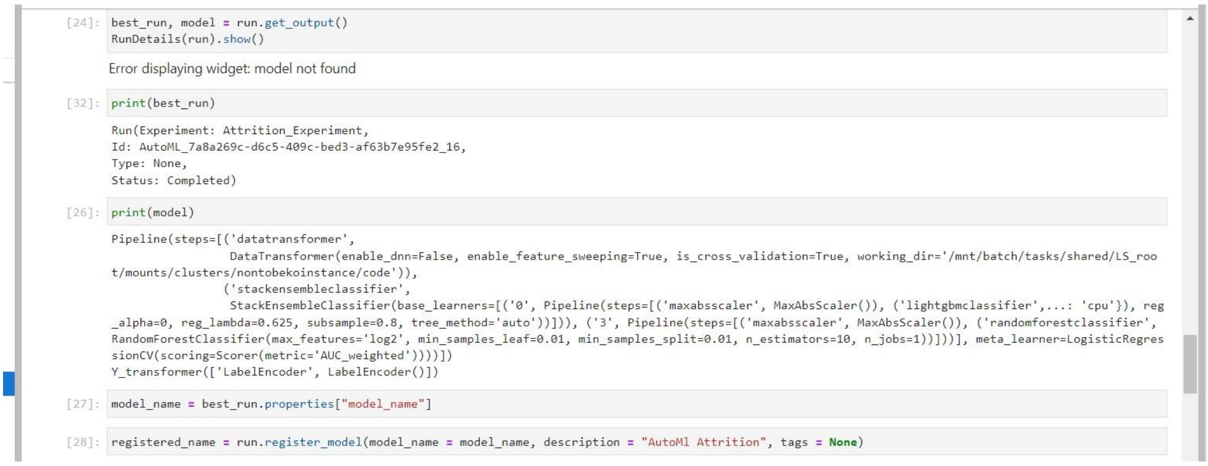


```
******************************************************************************************
DATA GUARDRAILS:

TYPE:         Class balancing detection
STATUS:       PASSED
DESCRIPTION:  Your inputs were analyzed, and all classes are balanced in your training data.
              Learn more about imbalanced data: https://aka.ms/AutomatedMLImbalancedData

******************************************************************************************

TYPE:         Missing feature values imputation
STATUS:       PASSED
DESCRIPTION:  No feature missing values were detected in the training data.
              Learn more about missing value imputation: https://aka.ms/AutomatedMLFeaturization

******************************************************************************************

TYPE:         High cardinality feature detection
STATUS:       PASSED
DESCRIPTION:  Your inputs were analyzed, and no high cardinality features were detected.
              Learn more about high cardinality feature handling: https://aka.ms/AutomatedMLFeaturization

******************************************************************************************
Current status: ModelSelection. Beginning model selection.

******************************************************************************************
ITER: The iteration being evaluated.
PIPELINE: A summary description of the pipeline being evaluated.
DURATION: Time taken for the current iteration.
METRIC: The result of computing score on the fitted pipeline.
BEST: The best observed score thus far.
******************************************************************************************
```



```
******************************************************************************************

 ITER   PIPELINE                                       DURATION            METRIC      BEST
    0   MaxAbsScaler LightGBM                          0:00:38             0.8292    0.8292
    1   MaxAbsScaler XGBoostClassifier                 0:00:54             0.8126    0.8292
    2   MaxAbsScaler ExtremeRandomTrees                0:00:38             0.7929    0.8292
    3   MaxAbsScaler RandomForest                      0:00:37             0.7947    0.8292
    4   StandardScalerWrapper LightGBM                 0:00:38             0.8110    0.8292
    5   SparseNormalizer XGBoostClassifier             0:00:51             0.8122    0.8292
    6   SparseNormalizer RandomForest                  0:00:43             0.8168    0.8292
    7   StandardScalerWrapper XGBoostClassifier        0:00:50             0.7798    0.8292
    8   SparseNormalizer XGBoostClassifier             0:00:50             0.8034    0.8292
    9   MaxAbsScaler RandomForest                      0:00:38             0.7717    0.8292
   10   SparseNormalizer LightGBM                      0:00:38             0.8077    0.8292
   11   MaxAbsScaler ExtremeRandomTrees                0:00:51             0.8255    0.8292
   12   StandardScalerWrapper XGBoostClassifier        0:00:50             0.8252    0.8292
   13   StandardScalerWrapper ExtremeRandomTrees       0:00:38             0.8153    0.8292
   14   StandardScalerWrapper RandomForest             0:00:55             0.8169    0.8292
   15   VotingEnsemble                                 0:00:50             0.8360    0.8360
   16   StackEnsemble                                  0:00:56             0.8372    0.8372
Stopping criteria reached at iteration 17. Ending experiment.
******************************************************************************************
Current status: BestRunExplainModel. Best run model explanations started
Current status: ModelExplanationDataSetSetup. Model explanations data setup completed
Current status: PickSurrogateModel. Choosing LightGBM as the surrogate model for explanations
Current status: EngineeredFeatureExplanations. Computation of engineered features started
2024-07-28:17:00:39,21 INFO     [explanation_client.py:334] Using default datastore for uploads
Current status: EngineeredFeatureExplanations. Computation of engineered features completed
Current status: RawFeaturesExplanations. Computation of raw features started
Current status: RawFeaturesExplanations. Computation of raw features completed
Current status: BestRunExplainModel. Best run model explanations completed
******************************************************************************************
```

Getting the best run output



```
[24]: best_run, model = run.get_output()
      RunDetails(run).show()

      Error displaying widget: model not found

[32]: print(best_run)

      Run(Experiment: Attrition_Experiment,
      Id: AutoML_7a8a269c-d6c5-409c-bed3-af63b7e95fe2_16,
      Type: None,
      Status: Completed)

[26]: print(model)

      Pipeline(steps=[('datatransformer',
                       DataTransformer(enable_dnn=False, enable_feature_sweeping=True, is_cross_validation=True, working_dir='/mnt/batch/tasks/shared/LS_roo
      t/mounts/clusters/nontobekoinstance/code')),
                      ('stackensembleclassifier',
                       StackEnsembleClassifier(base_learners=[('0', Pipeline(steps=[('maxabsscaler', MaxAbsScaler()), ('lightgbmclassifier',...: 'cpu'}), reg
      _alpha=0, reg_lambda=0.625, subsample=0.8, tree_method='auto'))])), ('3', Pipeline(steps=[('maxabsscaler', MaxAbsScaler()), ('randomforestclassifier',
      RandomForestClassifier(max_features='log2', min_samples_leaf=0.01, min_samples_split=0.01, n_estimators=10, n_jobs=1))]))], meta_learner=LogisticRegres
      sionCV(scoring=Scorer(metric='AUC_weighted'))))])
      Y_transformer(['LabelEncoder', LabelEncoder()])

[27]: model_name = best_run.properties["model_name"]

[28]: registered_name = run.register_model(model_name = model_name, description = "AutoMl Attrition", tags = None)
```

Registering the model

```
[27]: model_name = best_run.properties["model_name"]

[28]: registered_name = run.register_model(model_name = model_name, description = "AutoMl Attrition", tags = None)
```

```
[29]: from azureml.core.model import InferenceConfig
      from azureml.core.webservice import AciWebservice, Webservice
      from azureml.core.model import Model
      from azureml.core.environment import Environment
```

```
[30]: #Download the scoring files
      best_run.download_file("outputs/scoring_file_v_1_0_0.py", "inference/score.py")
```

Wait for the deployment to complete

```
[34]: from azureml.automl.core.shared import constants

      best_run.download_file(constants.CONDA_ENV_FILE_PATH, "myenvy.yml")
      env = Environment.from_conda_specification(name="myenvy", file_path = "myenvy.yml")

      inference_config = InferenceConfig(entry_script = "inference/score.py", environment=env)
      aciconfig = AciWebservice.deploy_configuration(cpu_cores = 1, memory_gb = 1 ,description = "attrition classification")
      service = Model.deploy(ws, "attrition", [registered_name], inference_config, aciconfig)

      service.wait_for_deployment(True)
```

Model Evaluation

Once trained, the model's performance is evaluated using metrics such as accuracy, precision, recall, and F1-score on the test dataset to assess its effectiveness in making predictions.

Procedure for testing the deployed model to ensure it performs as expected:

(a) Environment Configuration

(b) Data Preparation

(c) Input Data Validation

(d) Testing (Use typical examples of data that the model is expected to encounter in production)

(e) Prediction Output & Accuracy Assessment

(f) Performance Testing (Latency & Throughput

(g) Integration Testing

(h) Validation Against Baselines

(i) Bias and Fairness Testing (if applicable)

(j) Documentation of Testing Results

(k) Based on testing results, iteratively refine the model if necessary, addressing any identified issues or performance gaps.

Monitoring and Logging

Monitoring the performance and health of a deployed model is crucial for ensuring it continues to operate effectively and meets service level expectations. Azure provides several tools and services that can be leveraged for performance monitoring.

Azure Monitor

Metrics - Collects performance metrics such as CPU usage, memory usage, and response times of the deployed model endpoint.

Alerts - Set up alerts based on predefined thresholds for metrics, for example if response time exceeds a certain limit.

Logs: Azure Monitor can also collect logs from various Azure services, including Application Insights and Azure Machine Learning, to provide deeper insights into model performance.

Logging mechanisms are essential for troubleshooting, debugging, and auditing purposes.

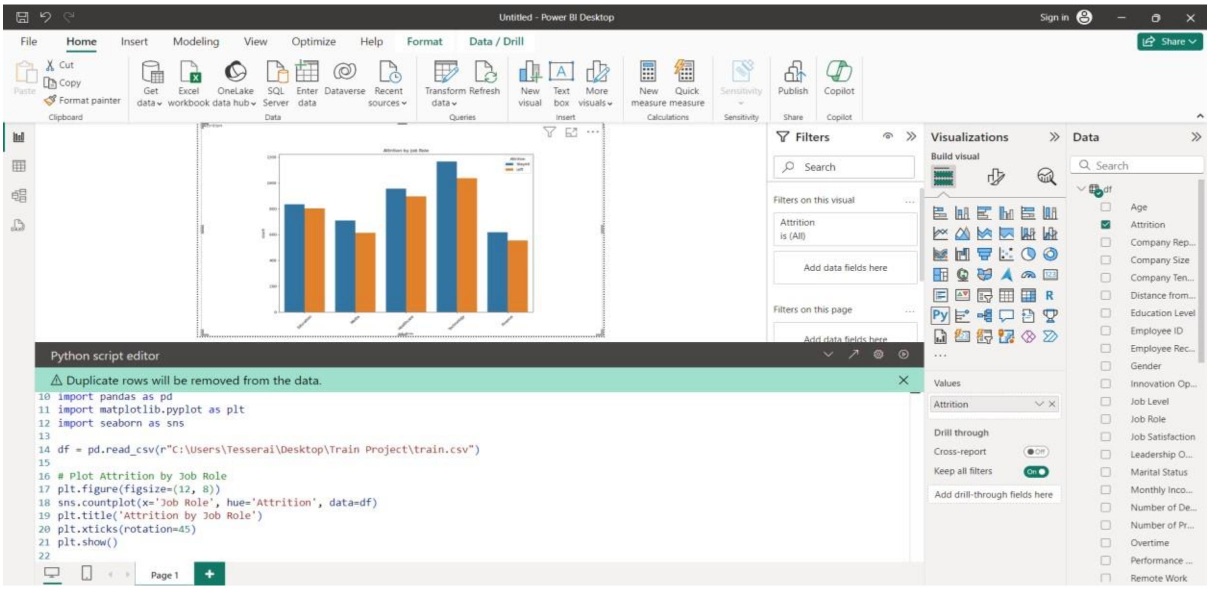
Azure Monitor Logs

Querying Logs - Use Azure Monitor Logs to query and analyse logs collected from various Azure services, including Application Insights and Azure Machine Learning.

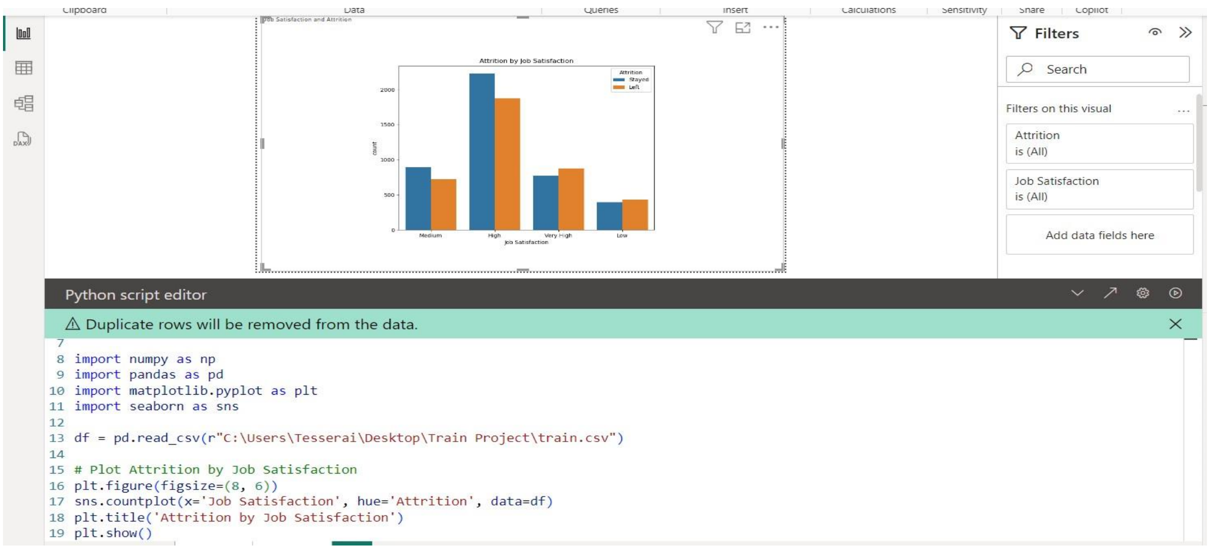
Log Analytics - Leverage Log Analytics to perform advanced queries, create dashboards, and gain insights into the operational health of the deployed model.

Data Visualization using PowerBI

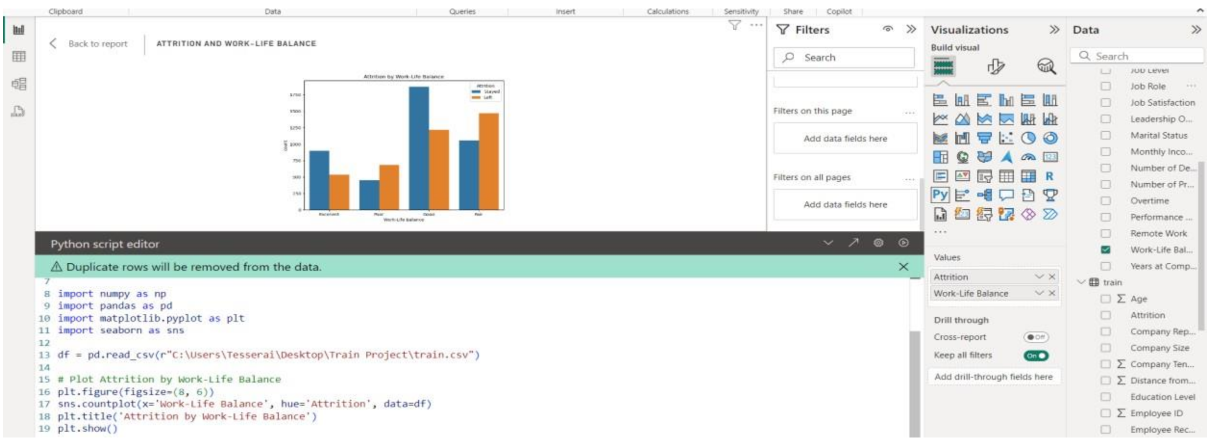
Plot Attrition by job role



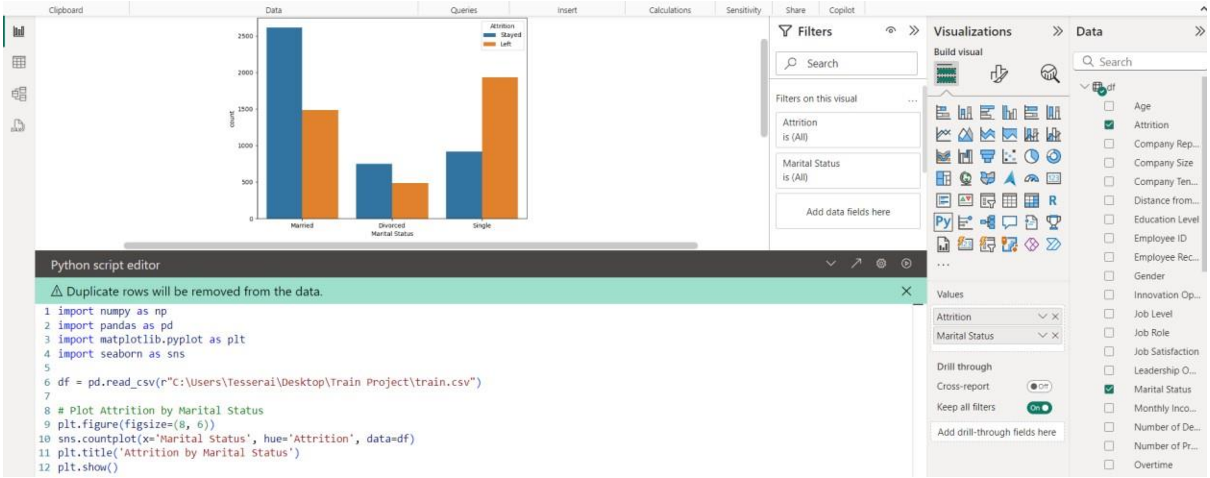
Plot Attrition by job satisfaction



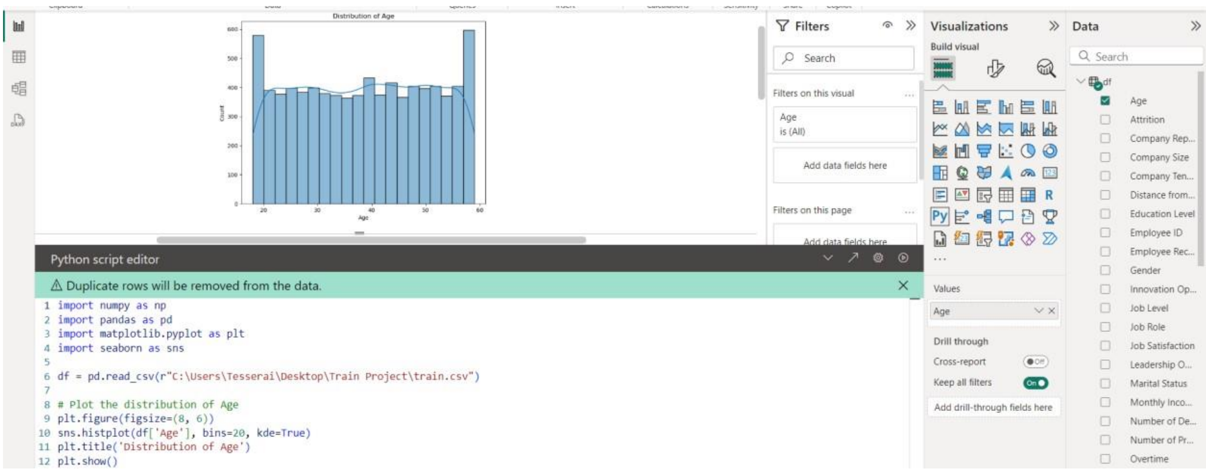
Plot Attrition by Work-Life balance



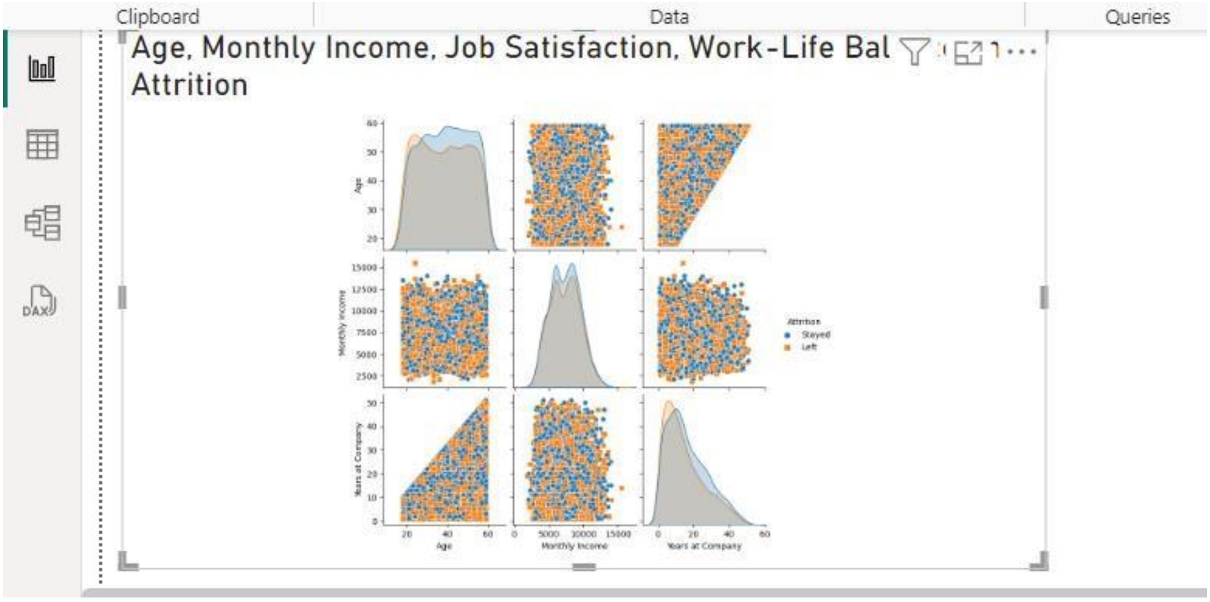
Plot attrition by marital status



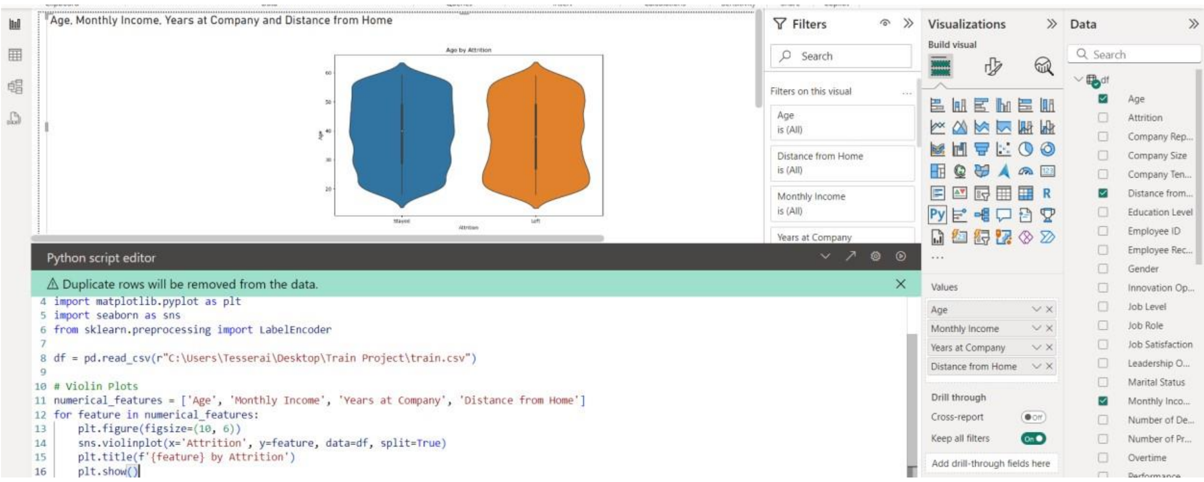
Plot the distribution of age



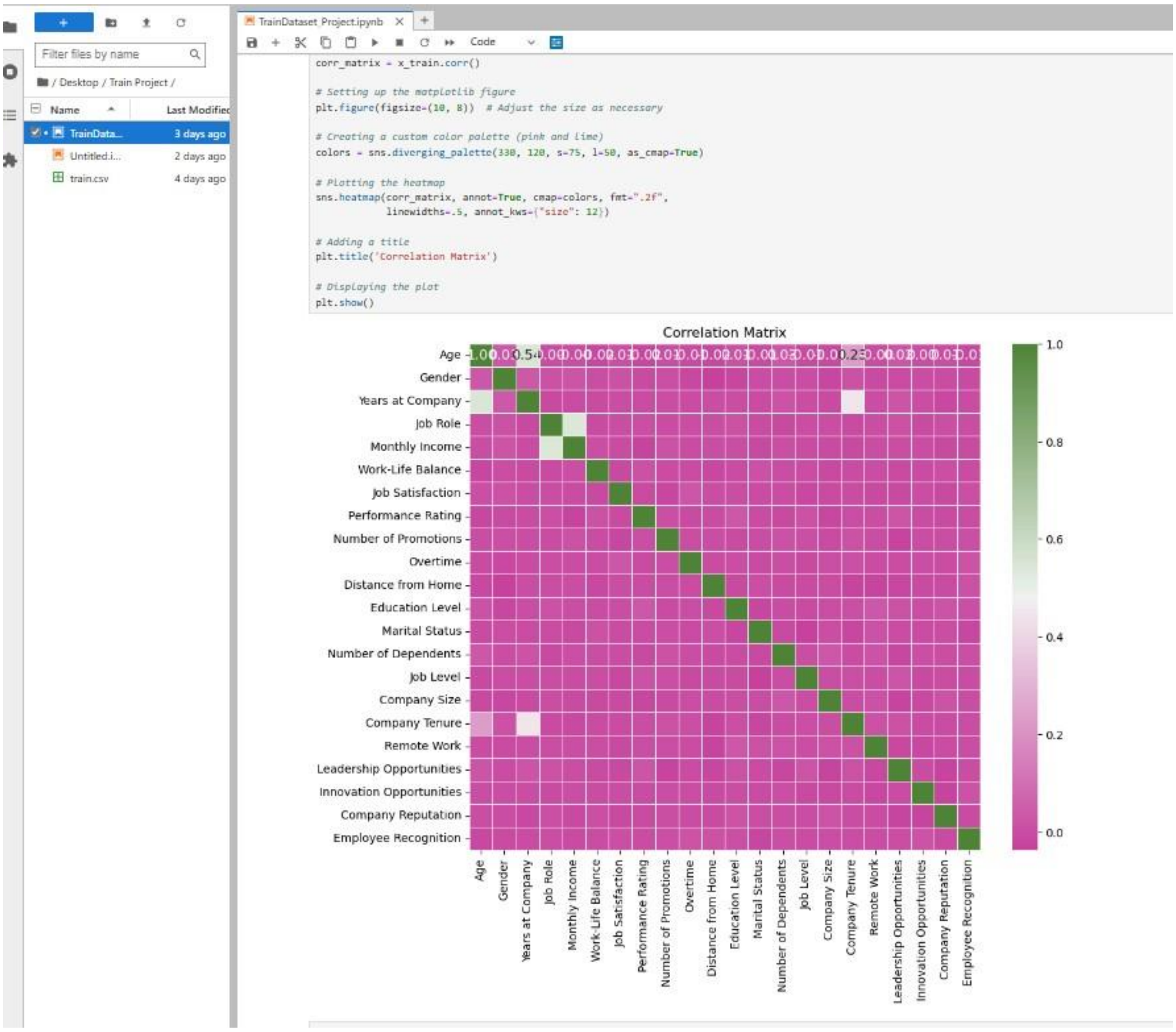
The confusion matrix



Violin plots



Hitmap



Data distribution

Monitoring and Maintenance:

Once deployed, the model's performance needs to be monitored regularly to ensure that it continues to provide accurate predictions over time. This may involve updating the model with new data or retraining it periodically to maintain its accuracy.

Deployment Environment used for the deployment of the model:

Frameworks and Libraries:

NumPy: For numerical computations and array manipulations.

Pandas: For data manipulation and analysis, particularly useful for handling datasets like the Iris dataset.

Scikit-learn: For machine learning algorithms and model training. It includes logistic regression and utilities for model evaluation.

Development Tools:

IDEs: PowerBI, VS Code, or Jupyter notebook for coding and testing.

Version Control: Git for managing code versions.

Package Management: pip or conda for managing Python packages and dependencies.

Security Considerations:

Access Control: Restrict access to the deployed model and its endpoints. Implement role-based access control (RBAC) to ensure only authorized personnel can interact with the model.

Encryption: Use encryption mechanisms (e.g., HTTPS/TLS) to secure data transmission between clients and the deployed model, preventing eavesdropping and data tampering.

Input Validation: Validate input data to prevent injection attacks and ensure that only expected and sanitized data is processed by the model.