

PROJECT OVERVIEW



PROJECT CARD

GOAL:

- *Build, train, test and deploy a machine learning classifier model to predict diabetes in patients*

TOOL:

- *AutoGluon*

PRACTICAL REAL-WORLD APPLICATION:

- *This project can be effectively used by healthcare professionals to detect diabetes and understand key factors that contribute to the disease.*

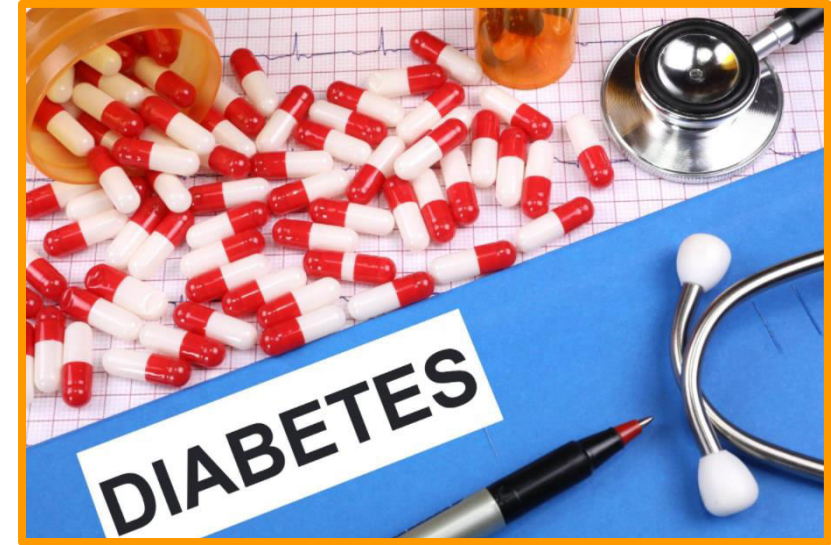
DATA:

• **INPUTS:**

- Pregnancies: Number of times pregnant
- GlucosePlasma: glucose concentration a 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- Skin: ThicknessTriceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (μ U/ml)
- BMI: Body mass index ($\text{weight in kg}/(\text{height in m})^2$)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)

• **OUTPUT:**

- Diabetes or no diabetes (0 or 1)



<https://www.flickr.com/photos/pasa/6757993805>

<https://www.kaggle.com/ljanjughazyan/cars1>

PROJECT OVERVIEW

- This dataset is used to predict whether or not a patient has diabetes, based on given features/diagnostic measurements.
- Only female patients are considered with at least 21 years old of Pima Indian heritage.



- Acknowledgements: Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261--265). IEEE Computer Society Press.
- Image Source: <https://iconscout.com/icon/doctor-1659516>

DATA OVERVIEW

The available features are:

- Pregnancies: Number of times pregnant
- GlucosePlasma: glucose concentration a 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- Skin: ThicknessTriceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)²)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)

Target (output):

- Diabetes or no diabetes (0 or 1)

Acknowledgements

Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261--265). IEEE Computer Society Press.

DATA OVERVIEW

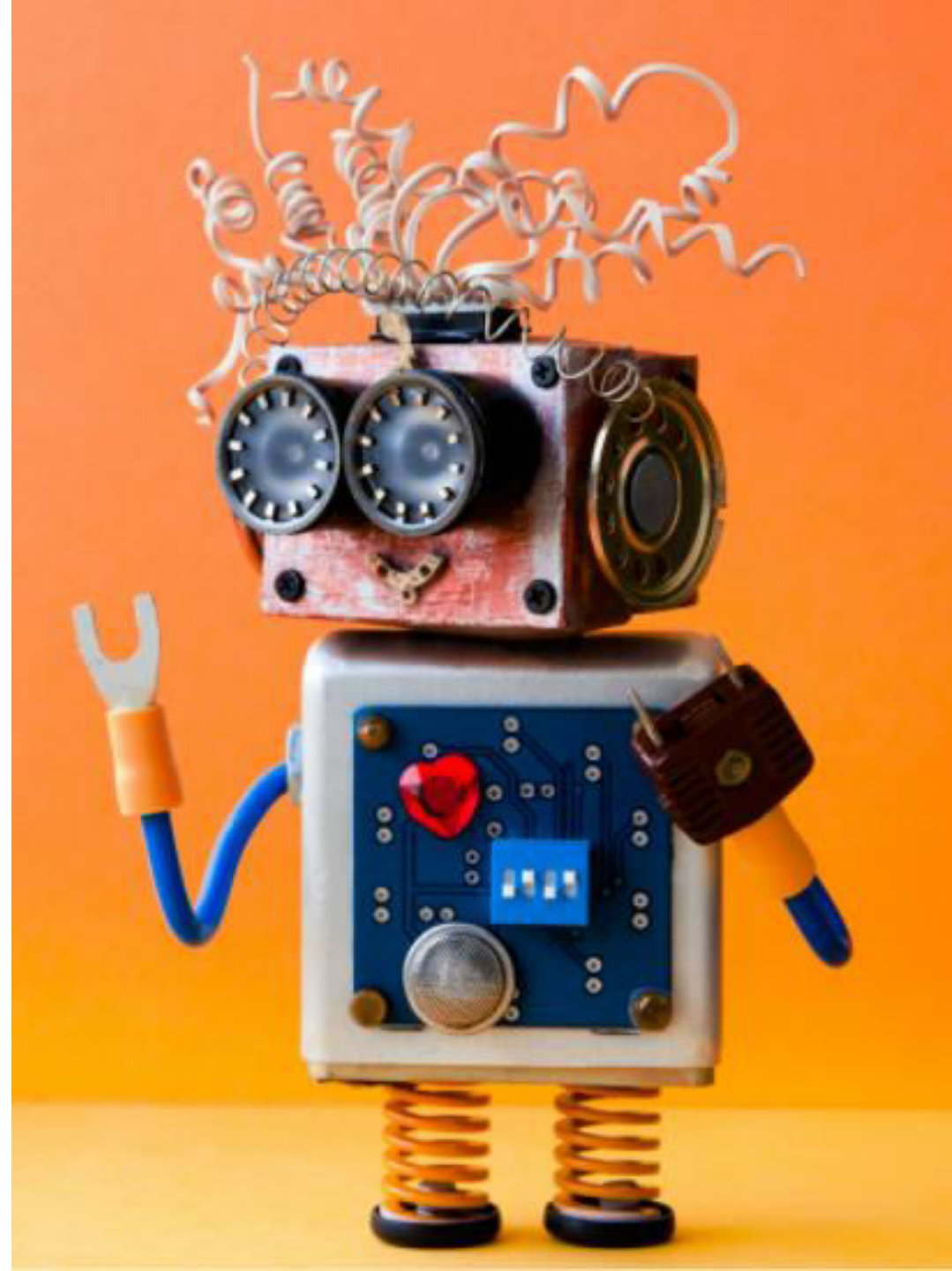
TARGET CLASS



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

CLASSIFICATION MODELS KPIs [SKIP IF FAMILIAR]

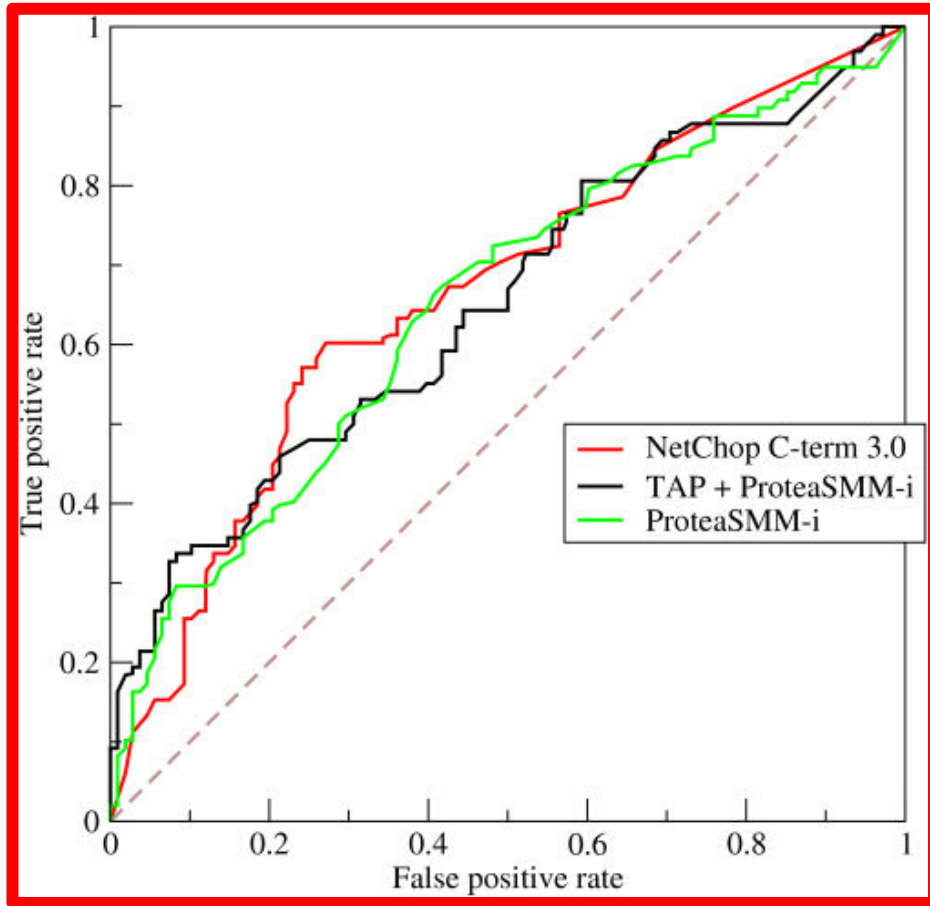


CLASSIFICATION MODEL KPIs

- Classification Accuracy = $(TP+TN) / (TP + TN + FP + FN)$
- Misclassification rate (Error Rate) = $(FP + FN) / (TP + TN + FP + FN)$
- Precision = $TP / \text{Total TRUE Predictions} = TP / (TP+FP)$ (When model predicted TRUE class, how often was it right?)
- Recall = $TP / \text{Actual TRUE} = TP / (TP+FN)$ (when the class was actually TRUE, how often did the classifier get it right?)

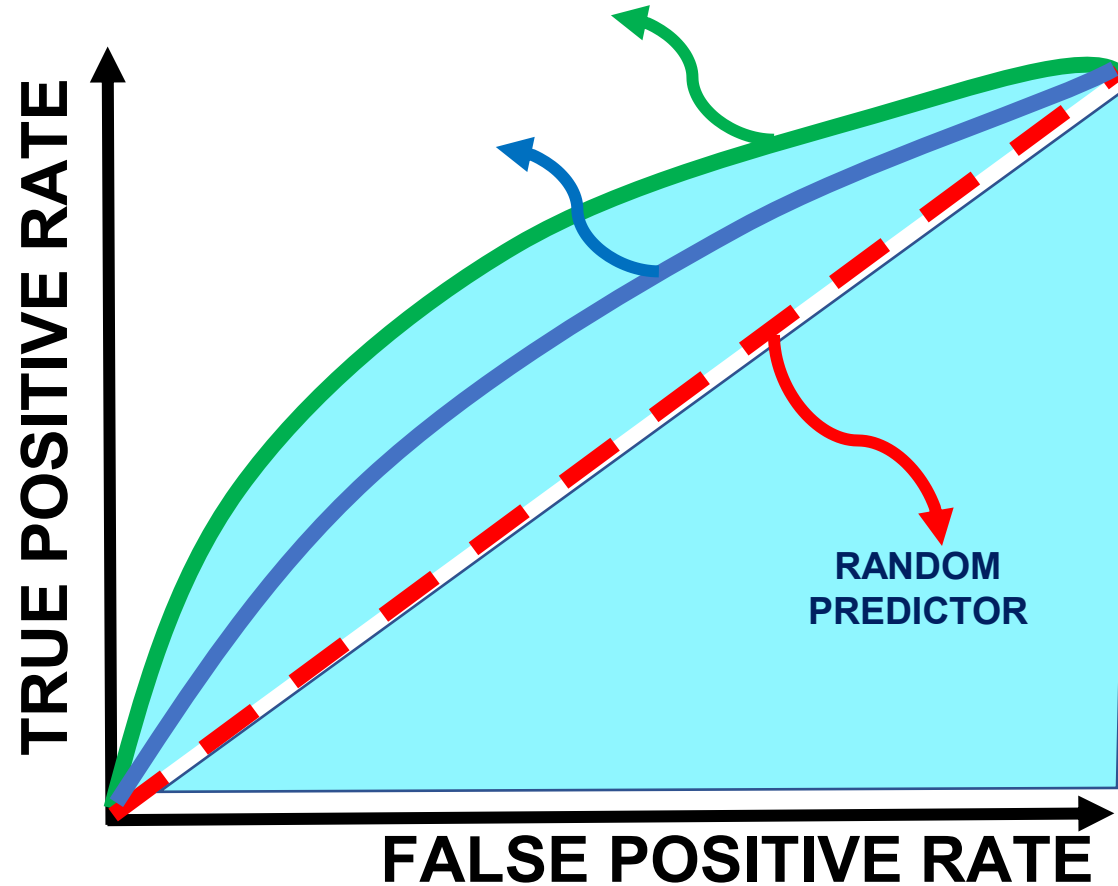
TRUE CLASS	
+	-
PREDICTIONS	
+	TRUE + FALSE +
-	FALSE - TRUE -

ROC (RECEIVER OPERATING CHARACTERISTIC CURVE)



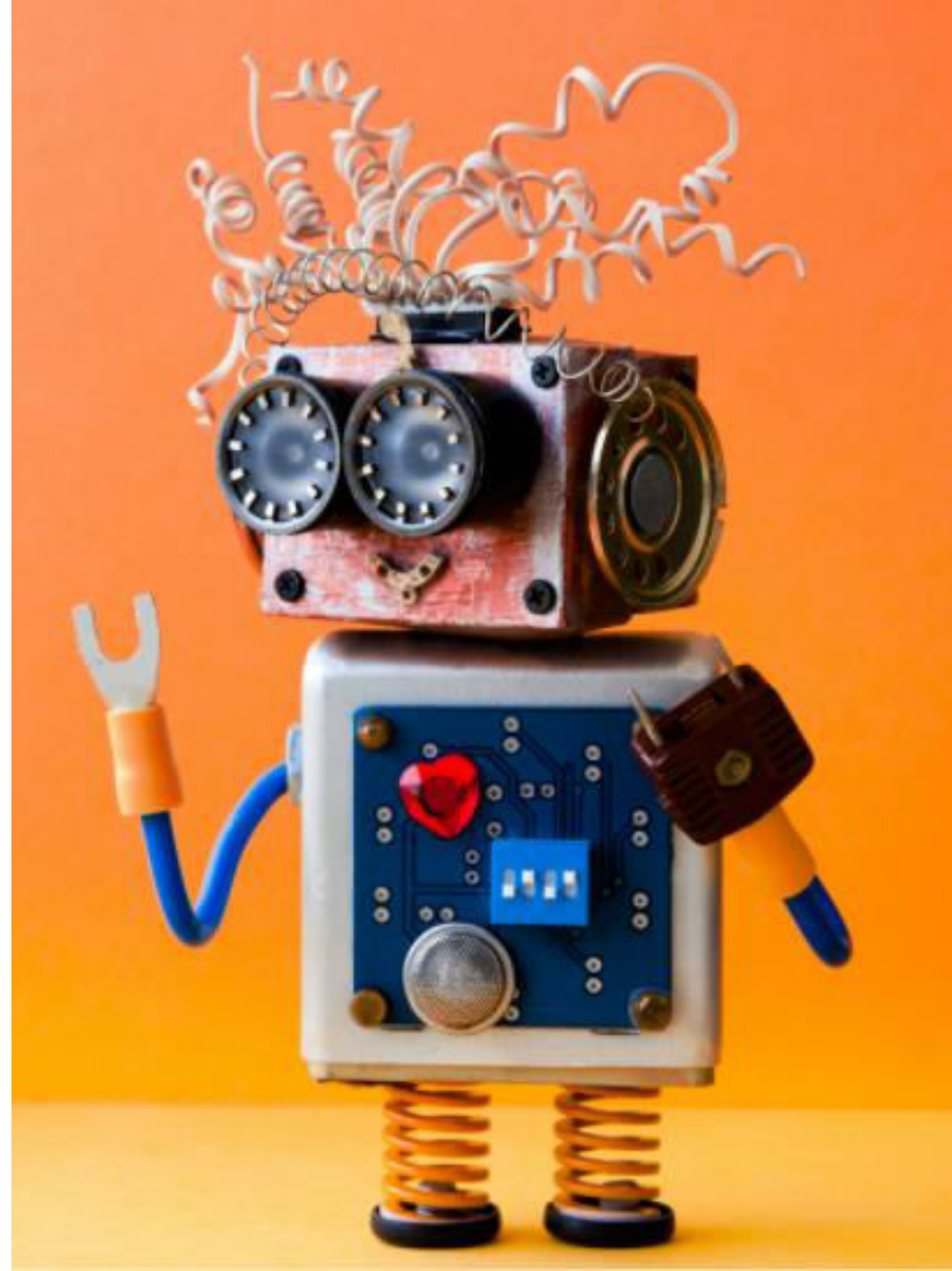
- ROC Curve is a metric that assesses the model ability to distinguish between binary (0 or 1) classes.
- The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.
- The true-positive rate is also known as sensitivity, recall or probability of detection in machine learning.
- The false-positive rate is also known as the probability of false alarm and can be calculated as $(1 - \text{specificity})$.
- Points above the diagonal line represent good classification (better than random)
- The model performance improves if it becomes skewed towards the upper left corner.

AUC (AREA UNDER CURVE)



- The light blue area represents the area Under the Curve of the Receiver Operating Characteristic (AUROC).
- The diagonal dashed red line represents the ROC curve of a random predictor with AUROC of 0.5.
- If ROC AUC = 1, perfect classifier
- Predictor #1 is better than predictor #2
- Higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s.

AUTOGLUON REVIEW [SKIP IF FAMILIAR]



AUTOGLUON 101

- AutoGluon allows for quick prototyping of AI/ML models using few simple lines of code.
- Autogluon works with text, image and tabular datasets.
- No need for expert level knowledge to train/test AI/ML models in Autogluon.
- Allows for automatic hyperparameters tuning and model selection.
- Check this out: <https://auto.gluon.ai/stable/index.html>
- All you need to do is to use these two lines of code!
- Please note that hyperparameters and all other model training aspects are set to default.



```
from autogluon.tabular import TabularPredictor  
predictor = TabularPredictor(label=<variable-name>).fit(train_data=<file-name>)
```

AUTOGLUON 101: TABULAR PREDICTOR PARAMETERS

- Check these out:
 - <https://auto.gluon.ai/stable/api/autogluon.task.html>
 - <https://auto.gluon.ai/stable/api/autogluon.task.html#autogluon.tabular.TabularPredictor.fit>

TabularPredictor

```
class autogluon.tabular. TabularPredictor (label, problem_type=None, eval_metric=None, path=None,
verbosity=2, sample_weight=None, weight_evaluation=False, groups=None, **kwargs)
```

[\[source\]](#)

- **Parameters:**
 - Label: Name of the column that contains the target variable to predict
 - problem_type, type of prediction problem, i.e. is this a binary/multiclass classification or regression problem (options: 'binary', 'multiclass', 'regression', 'quantile').
If problem_type = None, the prediction problem type is inferred based on the label-values in provided dataset.
 - eval_metric: test data metric, note that AutoGluon tunes hyperparameters and early-stopping to improve this metric on validation data.
 - If eval_metric = None, it is automatically chosen based on problem_type. Defaults to 'accuracy' for binary and multiclass classification, 'root_mean_squared_error' for regression, and 'pinball_loss' for quantile.
 - Otherwise, options for classification: ['accuracy', 'balanced_accuracy', 'f1', 'f1_macro', 'f1_micro', 'f1_weighted', 'roc_auc', 'roc_auc_ovo_macro', 'average_precision', 'precision', 'precision_macro', 'precision_micro', 'precision_weighted', 'recall', 'recall_macro', 'recall_micro', 'recall_weighted', 'log_loss', 'pac_score']
 - Options for regression: ['root_mean_squared_error', 'mean_squared_error', 'mean_absolute_error', 'median_absolute_error', 'r2']

AUTOGLUON 101: FIT METHOD PARAMETERS

```
fit (train_data, tuning_data=None, time_limit=None, presets=None, hyperparameters=None, feature_metadata='infer',  
      **kwargs) [source]
```

- Fit models to predict a column of a data table (label) based on the other columns (features).
- Parameters:
 - **train_data:** Table of the training data
 - **tuning_data:** validation data used for tuning processes such as early stopping and hyperparameter tuning. Don't include test data here!
 - **time_limit:** how long fit() should run for in seconds. If not specified, fit() will run until all models have completed training.
 - **Presets:** default = ['medium_quality_faster_train']
 - List of preset configurations for arguments in fit().
 - Presets impact predictive accuracy, memory-footprint, and inference latency of trained models
 - As an example, to get the most accurate overall predictor (regardless of its efficiency), set presets='best_quality'. To get good quality with minimal disk usage, set presets=['good_quality_faster_inference_only_refit', 'optimize_for_deployment']
 - Available Presets: ['best_quality', 'high_quality_fast_inference_only_refit', 'good_quality_faster_inference_only_refit', 'medium_quality_faster_train', 'optimize_for_deployment', 'ignore_text']

AUTOGLUON 101: PRESETS

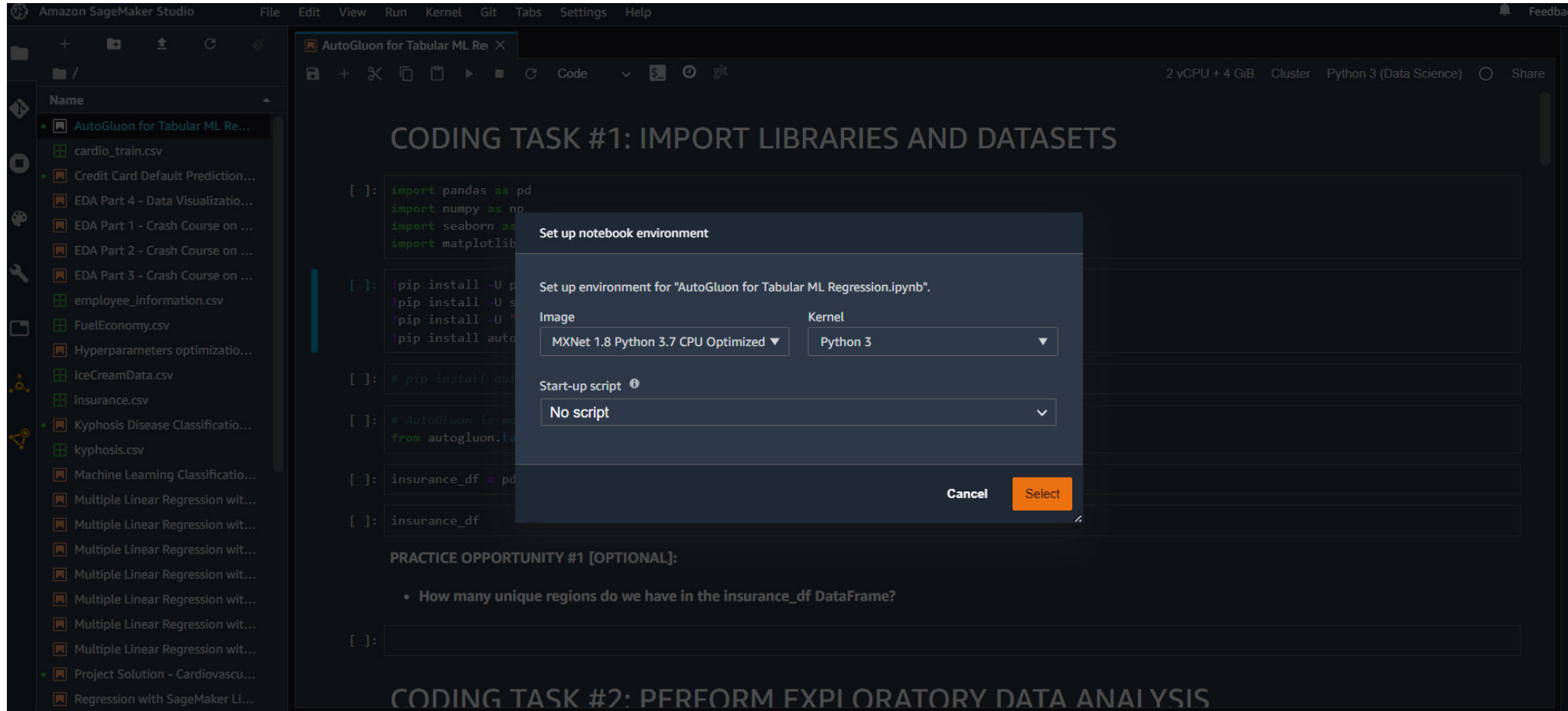
- **best_quality:** Best predictive accuracy with little consideration to inference time or disk usage.
- **high_quality_fast_inference_only_refit:** High predictive accuracy with fast inference. ~10x-200x faster inference and ~10x-200x lower disk usage than best_quality. Recommended for applications that require reasonable inference speed and/or model size.
- **good_quality_faster_inference_only_refit:** Good predictive accuracy with very fast inference. ~4x faster inference and ~4x lower disk usage than high_quality_fast_inference_only_refit. Recommended for applications that require fast inference speed.
- **medium_quality_faster_train:** Medium predictive accuracy with very fast inference and very fast training time. ~20x faster training than good_quality_faster_inference_only_refit. This is the default preset in AutoGluon, but should generally only be used for quick prototyping, as good_quality_faster_inference_only_refit results in significantly better predictive accuracy and faster inference time.
- **optimize_for_deployment:** Optimizes result immediately for deployment by deleting unused models and removing training artifacts. Often can reduce disk usage by ~2-4x with no negatives to model accuracy or inference speed. This will disable numerous advanced functionality, but has no impact on inference. This will make certain functionality less informative, such as `predictor.leaderboard()` and `predictor.fit_summary()`.

DEMO PART #1: CLASSIFICATION MODELS TRAINING USING AUTOGLUON



AUTOGLUON FOR CLASSIFICATION DEMO

CHOOSE THE IMAGE AND KERNEL SHOWN BELOW



The screenshot displays the Amazon SageMaker Studio interface. On the left, a file explorer shows a list of notebooks and data files, including 'AutoGluon for Tabular ML Regression.ipynb'. The main area shows a Jupyter notebook titled 'AutoGluon for Tabular ML Regression.ipynb' with the following code:

```
[ ]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

[ ]: !pip install -U pip
!pip install -U setuptools
!pip install -U autogluon.tabular

[ ]: # pip install autogluon.tabular

[ ]: # AutoGluon is installed
from autogluon.tabular import TabularPredictor

[ ]: insurance_df = pd.read_csv('insurance.csv')

[ ]: insurance_df
```

A modal dialog titled 'Set up notebook environment' is open, prompting the user to 'Set up environment for "AutoGluon for Tabular ML Regression.ipynb"'. The dialog includes the following options:

- Image:** MXNet 1.8 Python 3.7 CPU Optimized (selected)
- Kernel:** Python 3 (selected)
- Start-up script:** No script (selected)

The dialog has 'Cancel' and 'Select' buttons. Below the dialog, the notebook content shows a 'PRACTICE OPPORTUNITY #1 [OPTIONAL]:' section with a bullet point: 'How many unique regions do we have in the insurance_df DataFrame?'. The bottom of the notebook shows the heading 'CODING TASK #2: PERFORM EXPLORATORY DATA ANALYSIS'.

AUTOGLUON FOR CLASSIFICATION DEMO

```
[29]: # Train multiple ML classifier models using AutoGluon
# You need to specify the target column, train_data, limit_time, and presets
# Note that AutoGluon automatically detects if the problem is classification or regression type problems from the 'label' column

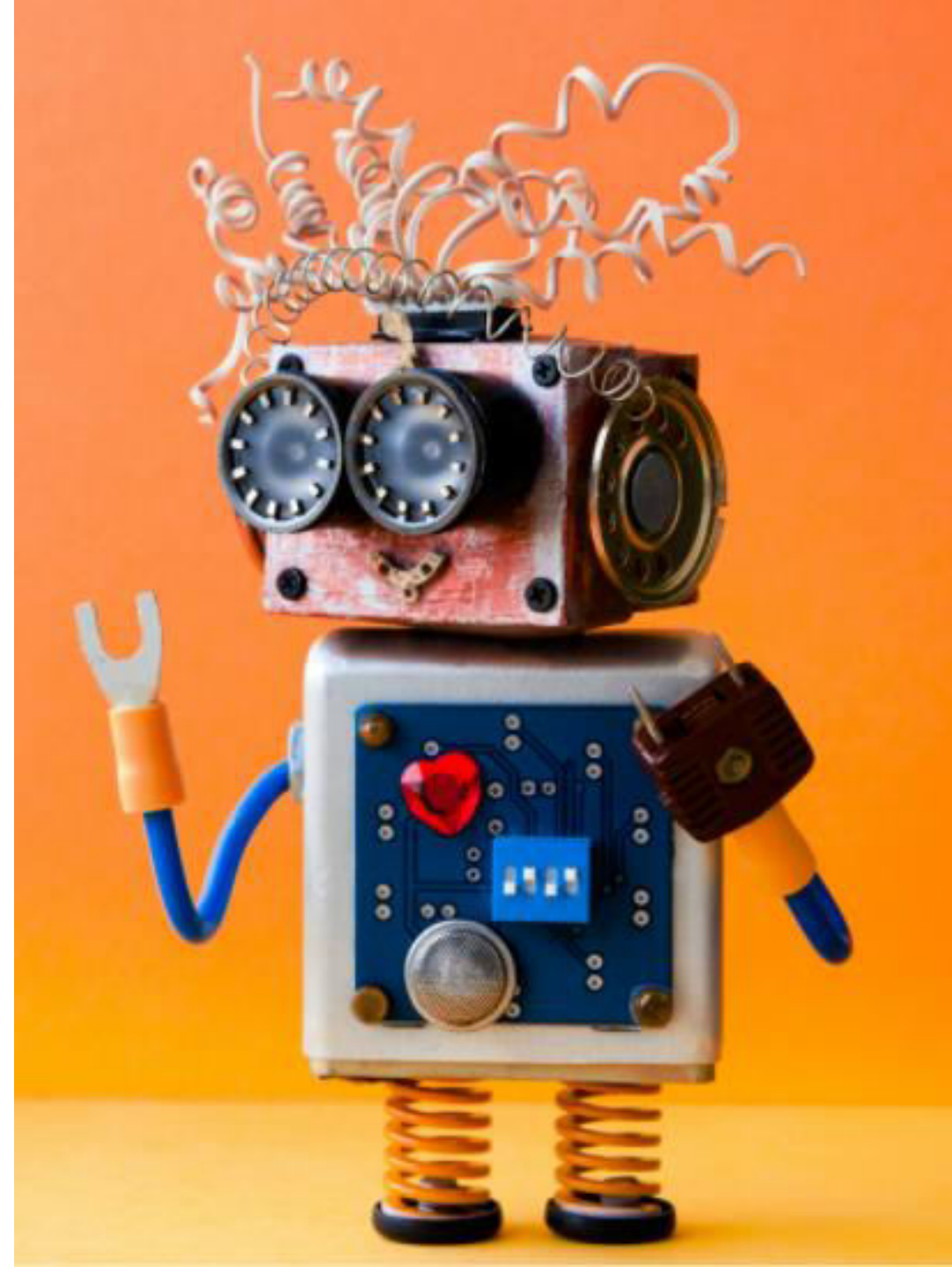
predictor = TabularPredictor(label="Outcome", problem_type = 'binary', eval_metric = 'accuracy').fit(train_data = X_train, time_limit = 200, presets =
```

```
No path specified. Models will be saved in: "AutogluonModels/ag-20220208_183244/"
Presets specified: ['best_quality']
Beginning AutoGluon training ... Time limit = 200s
AutoGluon will save models to "AutogluonModels/ag-20220208_183244/"
AutoGluon Version: 0.3.1
Train Data Rows: 614
Train Data Columns: 8
Preprocessing data ...
Selected class <--> label mapping: class 1 = 1, class 0 = 0
Using Feature Generators to preprocess the data ...
Fitting AutoMLPipelineFeatureGenerator...
    Available Memory: 2869.48 MB
    Train Data (Original) Memory Usage: 0.04 MB (0.0% of available memory)
    Inferring data type of each feature based on column values. Set feature_metadata_in to manually specify special dtypes of the features.
    Stage 1 Generators:
        Fitting AsTypeFeatureGenerator...
    Stage 2 Generators:
        Fitting FillNaFeatureGenerator...
    Stage 3 Generators:
        Fitting IdentityFeatureGenerator...
    Stage 4 Generators:
        Fitting DropUniqueFeatureGenerator...
    Types of features in original data (raw dtype, special dtypes):
        ('float', []) : 2 | ['BMI', 'DiabetesPedigreeFunction']
        ('int', []) : 6 | ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', ...]
    Types of features in processed data (raw dtype, special dtypes):
        ('float', []) : 2 | ['BMI', 'DiabetesPedigreeFunction']
        ('int', []) : 6 | ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', ...]
```

AUTOGLUON FOR CLASSIFICATION DEMO

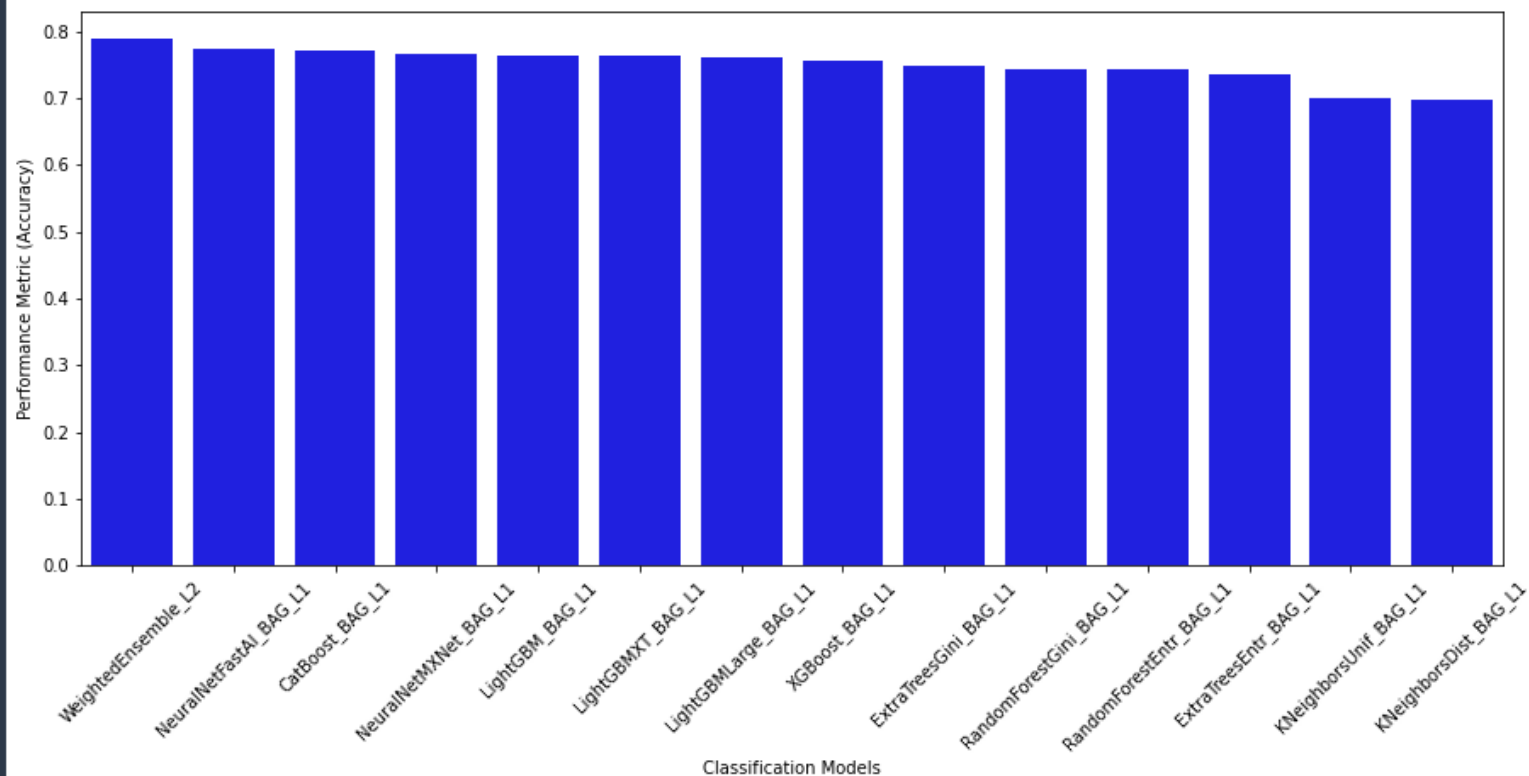
	model	score_val	pred_time_val	fit_time	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer	fit_order
0	WeightedEnsemble_L2	0.789902	7.275872	146.798918	0.001702	0.616199	2	True	14
1	NeuralNetFastAI_BAG_L1	0.773616	0.350929	20.404723	0.350929	20.404723	1	True	10
2	CatBoost_BAG_L1	0.771987	0.043355	8.469852	0.043355	8.469852	1	True	7
3	NeuralNetMXNet_BAG_L1	0.765472	6.641026	88.185018	6.641026	88.185018	1	True	12
4	LightGBM_BAG_L1	0.763844	0.066962	8.967226	0.066962	8.967226	1	True	4
5	LightGBMXT_BAG_L1	0.763844	0.072416	8.781063	0.072416	8.781063	1	True	3
6	LightGBMLarge_BAG_L1	0.762215	0.070630	14.778105	0.070630	14.778105	1	True	13
7	XGBoost_BAG_L1	0.755700	0.101268	5.377795	0.101268	5.377795	1	True	11
8	ExtraTreesGini_BAG_L1	0.747557	0.118607	0.639354	0.118607	0.639354	1	True	8
9	RandomForestGini_BAG_L1	0.744300	0.109096	0.806482	0.109096	0.806482	1	True	5
10	RandomForestEntr_BAG_L1	0.744300	0.114796	0.739419	0.114796	0.739419	1	True	6
11	ExtraTreesEntr_BAG_L1	0.734528	0.111371	0.737729	0.111371	0.737729	1	True	9
12	KNeighborsUnif_BAG_L1	0.700326	0.105286	0.013805	0.105286	0.013805	1	True	1
13	KNeighborsDist_BAG_L1	0.697068	0.102086	0.004064	0.102086	0.004064	1	True	2

DEMO PART #2: CLASSIFICATION MODELS EVALUATION USING AUTOGLUON



AUTOGLUON FOR CLASSIFICATION DEMO

	model	score_val	pred_time_val	fit_time	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer	fit_order
0	WeightedEnsemble_L2	0.789902	7.275872	146.798918	0.001702	0.616199	2	True	14
1	NeuralNetFastAI_BAG_L1	0.773616	0.350929	20.404723	0.350929	20.404723	1	True	10
2	CatBoost_BAG_L1	0.771987	0.043355	8.469852	0.043355	8.469852	1	True	7
3	NeuralNetMXNet_BAG_L1	0.765472	6.641026	88.185018	6.641026	88.185018	1	True	12
4	LightGBM_BAG_L1	0.763844	0.066962	8.967226	0.066962	8.967226	1	True	4
5	LightGBMXt_BAG_L1	0.763844	0.072416	8.781063	0.072416	8.781063	1	True	3
6	LightGBMLarge_BAG_L1	0.762215	0.070630	14.778105	0.070630	14.778105	1	True	13
7	XGBoost_BAG_L1	0.755700	0.101268	5.377795	0.101268	5.377795	1	True	11
8	ExtraTreesGini_BAG_L1	0.747557	0.118607	0.639354	0.118607	0.639354	1	True	8
9	RandomForestGini_BAG_L1	0.744300	0.109096	0.806482	0.109096	0.806482	1	True	5
10	RandomForestEntr_BAG_L1	0.744300	0.114796	0.739419	0.114796	0.739419	1	True	6
11	ExtraTreesEntr_BAG_L1	0.734528	0.111371	0.737729	0.111371	0.737729	1	True	9
12	KNeighborsUnif_BAG_L1	0.700326	0.105286	0.013805	0.105286	0.013805	1	True	1
13	KNeighborsDist_BAG_L1	0.697068	0.102086	0.004064	0.102086	0.004064	1	True	2



AUTOGLUON FOR CLASSIFICATION DEMO

```
[33]: predictor.evaluate(X_test)
```

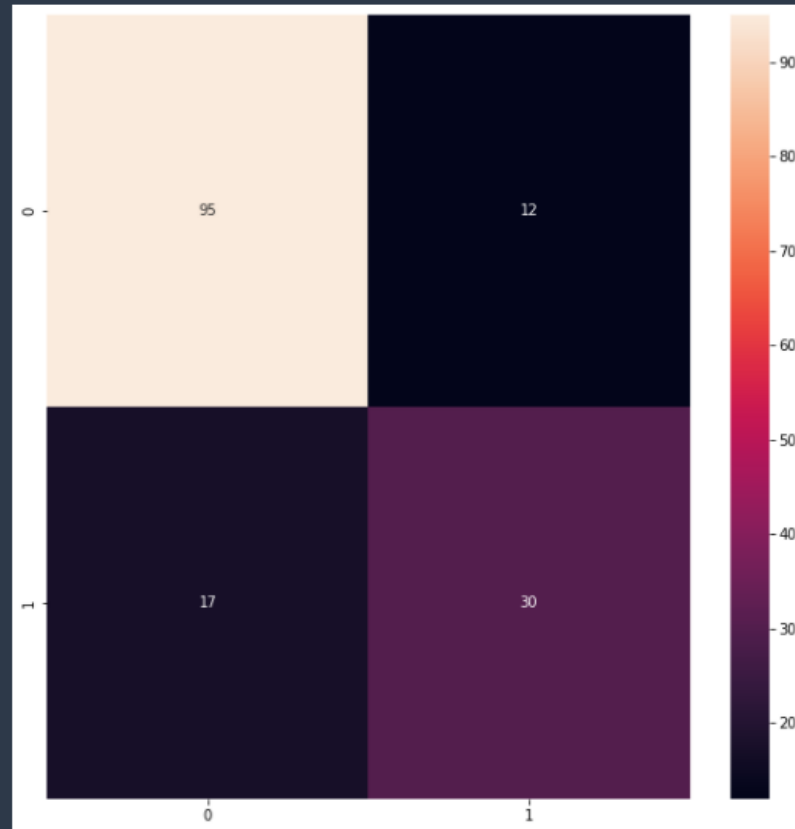
```
Evaluation: accuracy on test data: 0.8116883116883117
```

```
Evaluations on test data:
```

```
{  
  "accuracy": 0.8116883116883117,  
  "balanced_accuracy": 0.7630741698150726,  
  "mcc": 0.5440206255666752,  
  "roc_auc": 0.8627957844501889,  
  "f1": 0.6741573033707866,  
  "precision": 0.7142857142857143,  
  "recall": 0.6382978723404256  
}
```

```
[33]: {'accuracy': 0.8116883116883117,  
      'balanced_accuracy': 0.7630741698150726,  
      'mcc': 0.5440206255666752,  
      'roc_auc': 0.8627957844501889,  
      'f1': 0.6741573033707866,  
      'precision': 0.7142857142857143,  
      'recall': 0.6382978723404256}
```

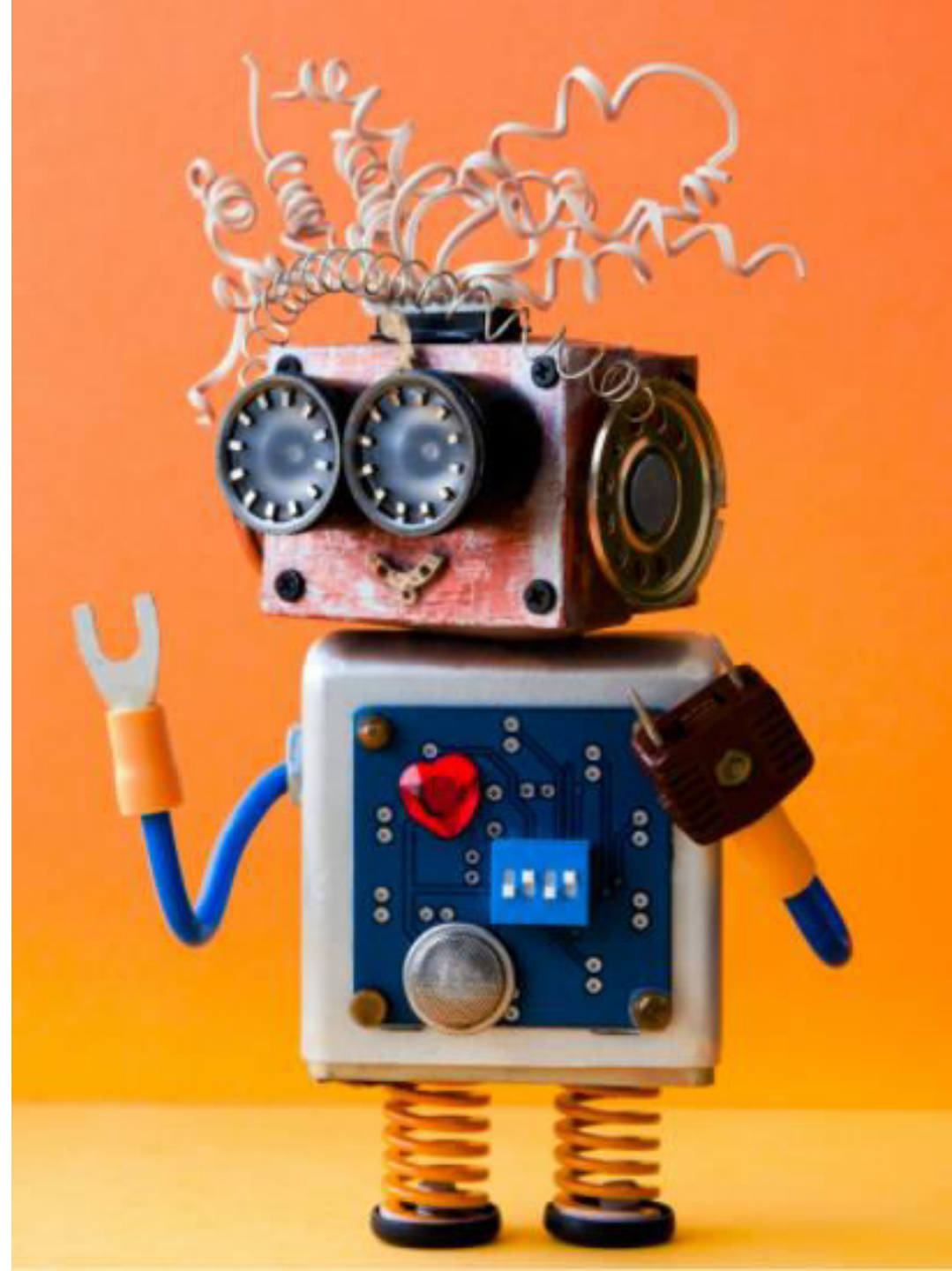
AUTOGLUON FOR CLASSIFICATION DEMO



```
[38]: from sklearn.metrics import classification_report  
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.85	0.89	0.87	107
1	0.71	0.64	0.67	47
accuracy			0.81	154
macro avg	0.78	0.76	0.77	154
weighted avg	0.81	0.81	0.81	154

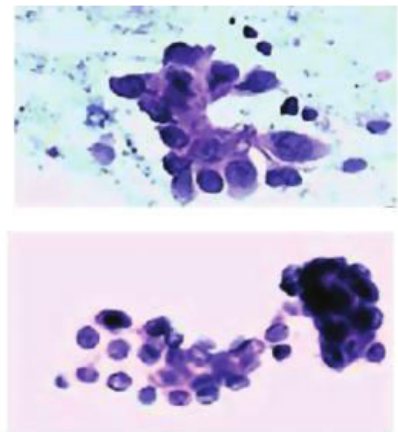
FINAL END-OF-DAY CAPSTONE PROJECT



PROJECT OVERVIEW

- Breast cancer is the most common cancer among women worldwide, accounting for 25% of all cancer cases and affected 2.1 Million people in 2015.
- Early diagnosis significantly increases the chances of survival. A key challenge in cancer detection is how to classify tumors into (1) malignant or (2) benign.
- Machine Learning (ML) dramatically improves the accuracy of diagnosis. Research indicates that most experienced physicians can diagnose cancer with 79.97% accuracy while 91.1% (higher) correct diagnosis is achieved using ML.

INPUT: 30 FEATURES



RADIUS
TEXTURE
PERIMETER
AREA
SMOOTHNESS

CLASSIFIER

TARGET CLASS: 2

MALIGNANT

BENIGN

Dataset:

- Number of Instances: 569
- Class Distribution: 212 Malignant, 357 Benign

PROJECT TASKS:

Please complete the following:

1. Load the “*cancer.csv*” dataset
2. Perform basic Exploratory Data Analysis
3. Split the data into 80% for training and 20% for testing
4. Using ‘best_quality’ preset and accuracy metric, train machine linear classification models using AutoGluon to predict the “class” column
5. Assess trained models performance by plotting the leaderboard and indicating the best model. Plot the confusion matrix.
6. Using ‘optimized for deployment’ preset and AUC metric, train machine linear classification models using AutoGluon.
7. Assess trained models performance by plotting the leaderboard and indicating the best model