

PROJECT OVERVIEW



PROJECT CARD

GOAL:

- The objective of this case study is to predict the health insurance cost incurred by Individuals based on their age, gender, BMI, number of children, smoking habit and geo-location.

TOOL:

- *AutoGluon (AutoGluon allows for quick prototyping of AI/ML models using few simple lines of code)*

PRACTICAL REAL-WORLD APPLICATION:

- *This project can be effectively used by insurance companies to predict healthcare insurance cost, increase revenues and reduce costs.*

DATA:

• **INPUTS:**

- age, gender, BMI, number of children, smoking habit and geo-location

• **OUTPUT:**

- *Insurance Charges*



Image source: <https://www.publicdomainpictures.net/en/view-image.php?image=279909&picture=medical-insurance>

DATA OVERVIEW

The available features are:

- sex: insurance contractor gender
- bmi: Body mass index (ideally 18.5 to 24.9)
- children: Number of children covered by health insurance / Number of dependents
- smoker: Smoking habits
- region: the beneficiary's residential area in the US, northeast, southeast, southwest, northwest.

Target (output):

- charges: Individual medical costs billed by health insurance

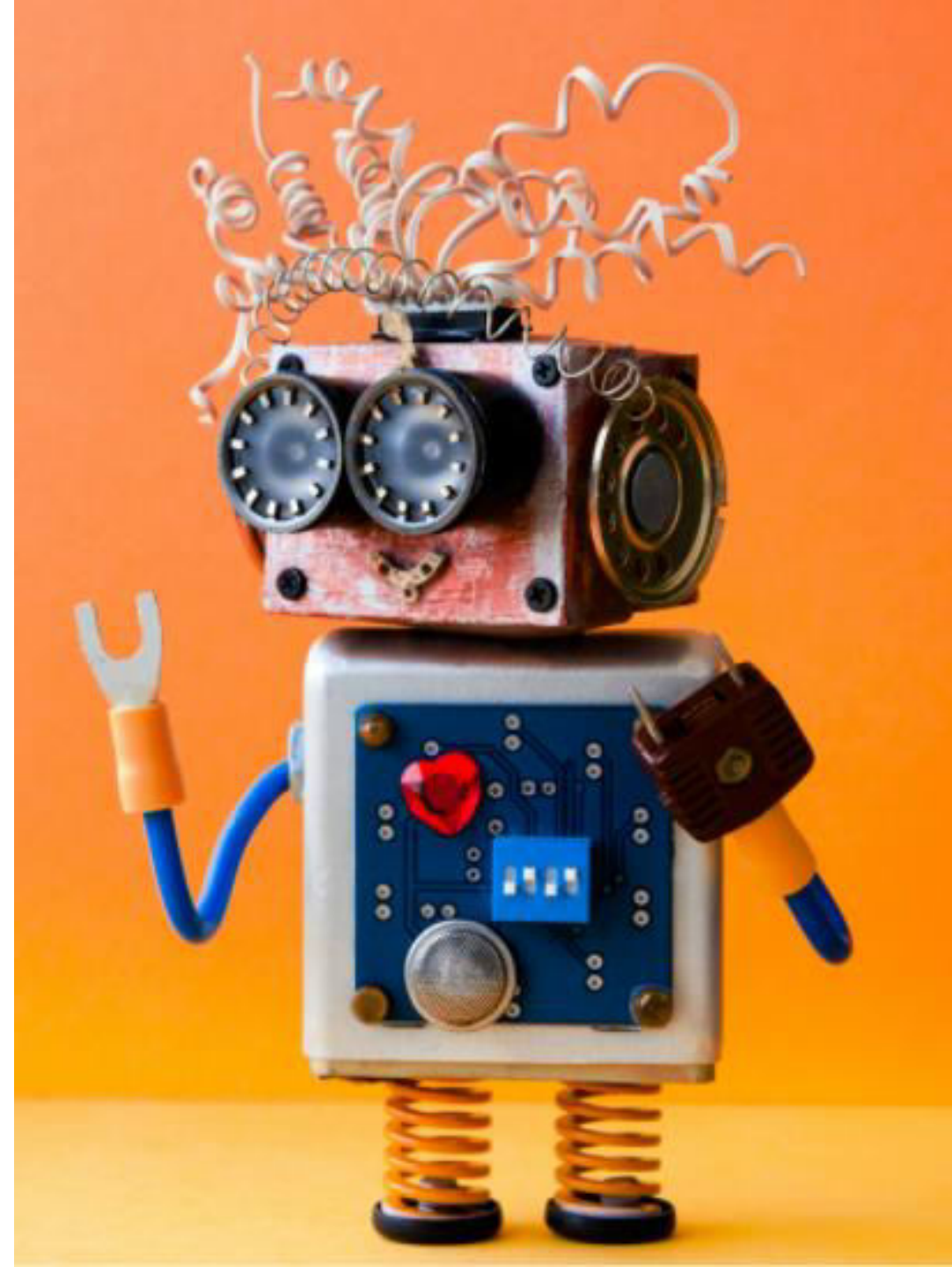
- Data Source: <https://www.kaggle.com/mirichoi0218/insurance>

DATA OVERVIEW

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030
1338 rows × 7 columns							

TARGET
COLUMN

AI/ML APPLICATIONS IN BUSINESS



AI IN BUSINESS: WHAT PROBLEMS CAN AI SOLVE?

CLUSTERING-TYPE PROBLEMS: “WHICH GROUP OF MY CUSTOMERS SHOULD I TARGET FOR MY NEW MARKETING AD CAMPAIGN TO MAXIMIZE CONVERSION RATE?”

- AI can help solve clustering-type problems that are crucial in performing marketing segmentation. Clustering works by trying to group customers based on their features.

RECOMMENDATION-TYPE PROBLEMS: “WHICH PRODUCT SHOULD I RECOMMEND TO MY CUSTOMERS TO MAXIMIZE REVENUE?”

- AI can help solve recommendation-type problems by collaboratively studying customer historical behavioural data. These algorithms are called recommender systems, which you might notice while using such as Netflix or Amazon.

CLASSIFICATION-TYPE PROBLEMS: “ARE MY CUSTOMERS HAPPY OR NOT?”

- As a data scientist, you can leverage data science to solve classification type problems. The answer to these problems can generally belong to two or more categories.

REGRESSION-TYPE PROBLEMS: “HOW MUCH SHOULD I PAY MY NEWLY HIRED EMPLOYEE?”

- This is a regression-type problem where we attempt to predict a continuous output (salary) that does not necessarily belong to a group or category.

TIME-SERIES FORECASTING PROBLEMS: “HOW MUCH REVENUE WILL MY BUSINESS GENERATE NEXT MONTH?”

- This is a time series forecasting problem in which we attempt to predict the future based on past historical data.

READING TIME & QUIZ: AI APPLICATIONS IN BUSINESS

- Please read the article below and answer the following quiz.
 - Link to Article: <https://www.forbes.com/sites/forbestechcouncil/2018/09/27/15-business-applications-for-artificial-intelligence-and-machine-learning/?sh=46e73b43579f>



15 MINS



10 MINS

PRACTICE OPPORTUNITY



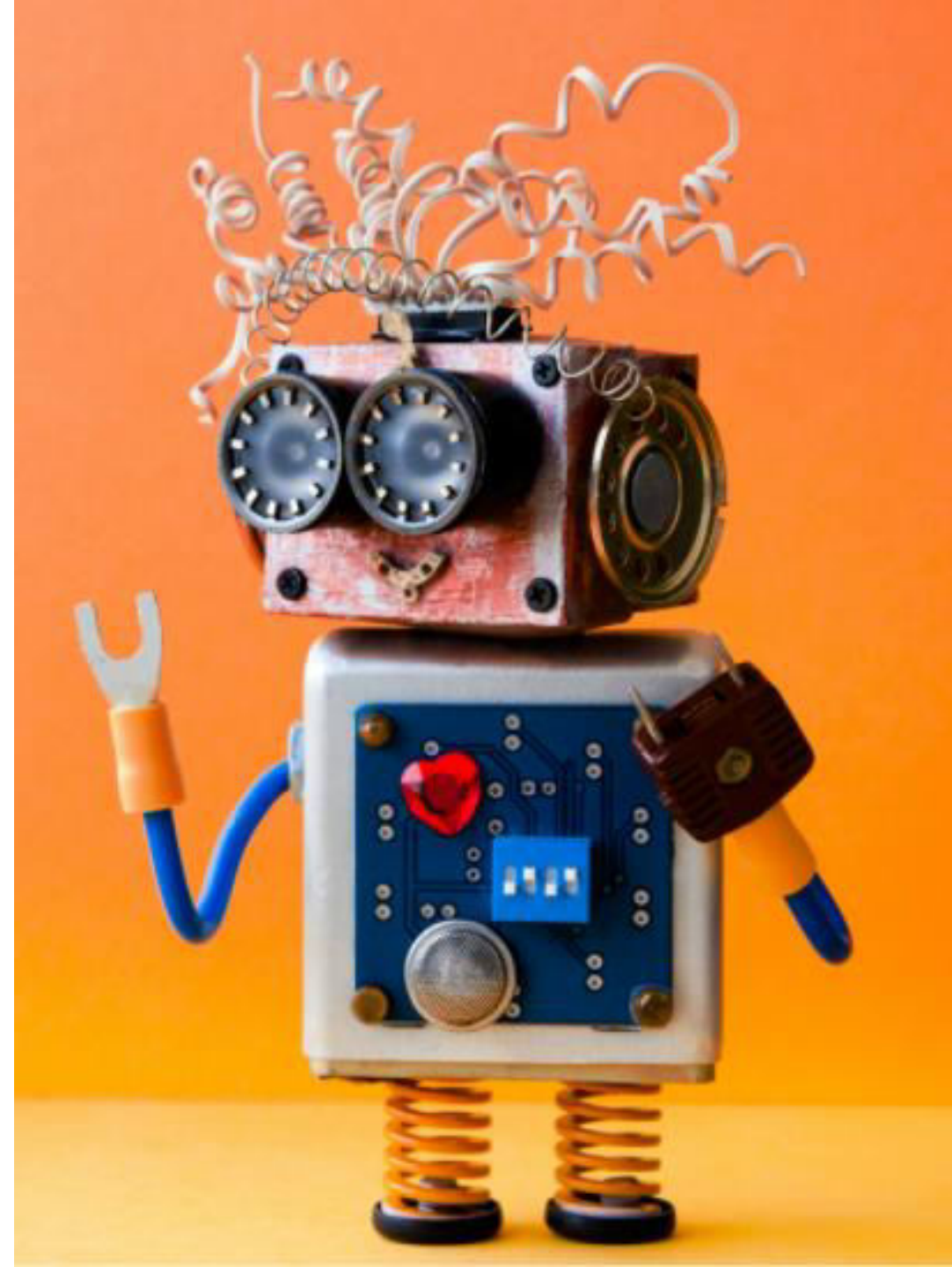
PRACTICE OPPORTUNITY

- Read this article by McKinsey & Company and answer the following questions:
<https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/the-executives-ai-playbook?page=industries/>
1. How much value does McKinsey & Company predict AI could potentially bring across all industries?
 2. What about the “insurance” industry?
 3. What about “risk modeling” domain?
 4. State the difference between traditional AI and Advanced AI in the article’s context.

PRACTICE OPPORTUNITY SOLUTION

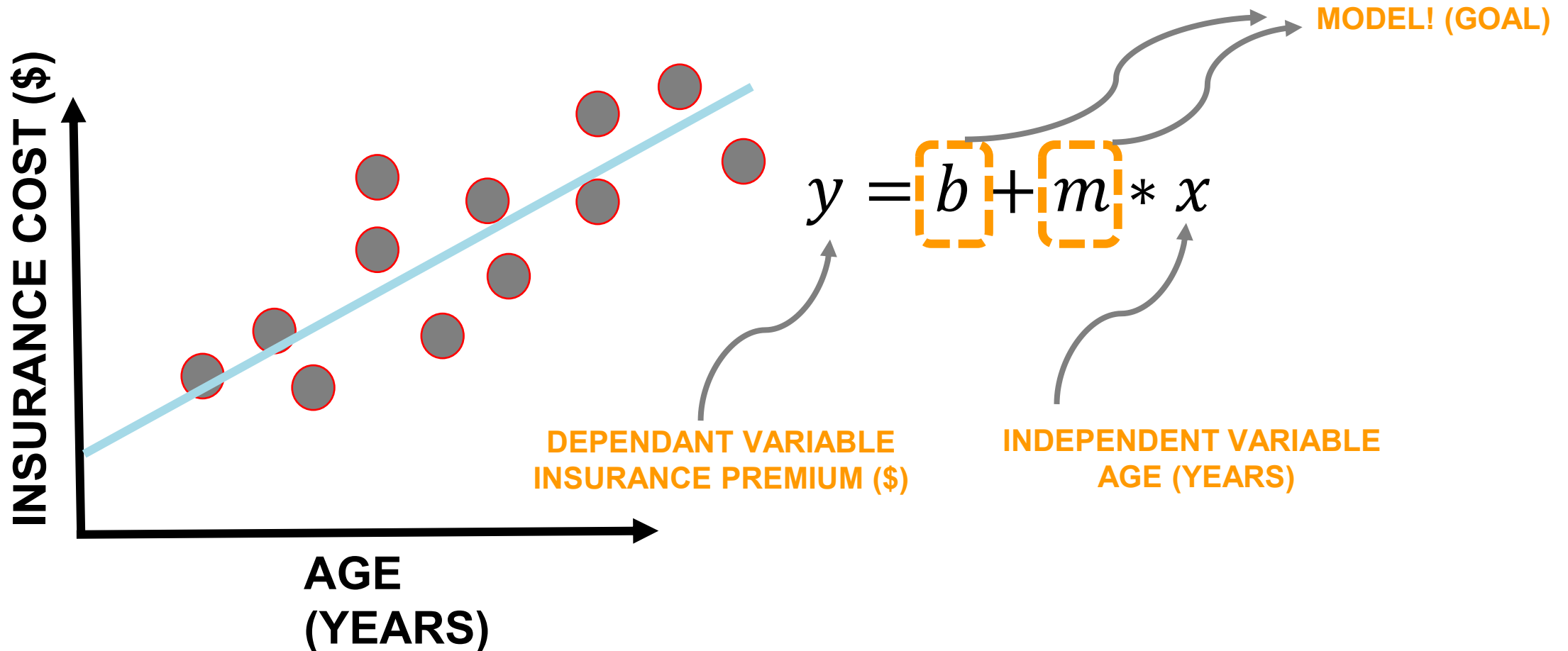
- Read this article by McKinsey & Company and answer the following questions:
<https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/the-executives-ai-playbook?page=industries/>
1. How much value does McKinsey & Company predict AI could potentially bring across all industries? **\$9.5T - \$15.4T**
 2. What about the “insurance” industry? **\$1.1T**
 3. What about “risk modeling” domain? **\$32.0B**
 4. State the difference between traditional AI and Advanced AI in the article’s context.
Traditional AI indicates basic machine learning and statistical techniques while advanced AI indicates deep learning and artificial neural networks.

REGRESSION REVIEW [SKIP IF FAMILIAR]



SIMPLE LINEAR REGRESSION

- Goal is to obtain a relationship (model) between two variables only such as age and insurance cost for example.



MULTIPLE LINEAR REGRESSION

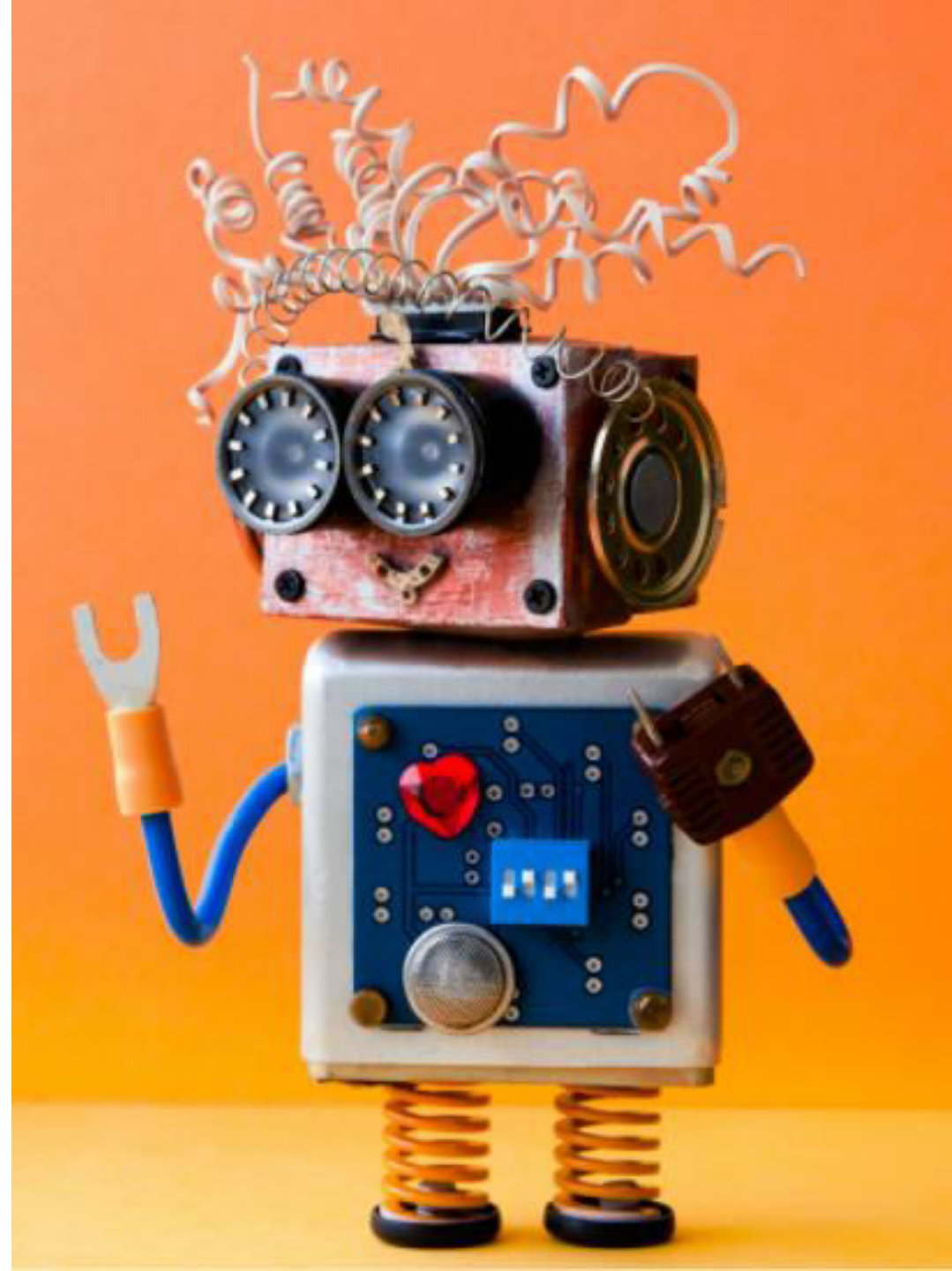
- Multiple Linear Regression: examines relationship between more than two variables.
- Recall that Simple Linear regression is a statistical model that examines linear relationship between two variables only.
- Each independent variable has its own corresponding coefficient.

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \dots + b_n x_n$$

DEPENDANT VARIABLES
INSURANCE COST (\$)

INDEPENDENT VARIABLES
(AGE, SMOKING HABITS,
REGION,..ETC)

WHAT IS AUTOGLUON?



AUTOGLUON 101

- In January 2020, AWS launched an open-source library called AutoGluon the library behind Sagemaker Autopilot.
- 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>
- Excellent Article: <https://aws.amazon.com/blogs/opensource/machine-learning-with-autogluon-an-open-source-automl-library/>
- Excellent Article 2: <https://www.philschmid.de/getting-started-with-automl-and-aws-autogluon>



AUTOGLUON 101

- AutoGluon is modularized into sub-modules for:
 1. Tabular
 2. text
 3. Images
- Install a sub-module using:


python3 -m pip install <submodule>

- **<submodule>** may be one of the following options:
 - autogluon.tabular - tabular data (TabularPredictor)
 - autogluon.vision - computer vision (ImagePredictor, ObjectDetector)
 - autogluon.text - natural language processing (TextPredictor)
 - autogluon.core - only core functionality (example: hyperparameter tuning of arbitrary code/models).
 - autogluon.features - feature generation/preprocessing pipelines (Tabular data).

AUTOGLUON 101

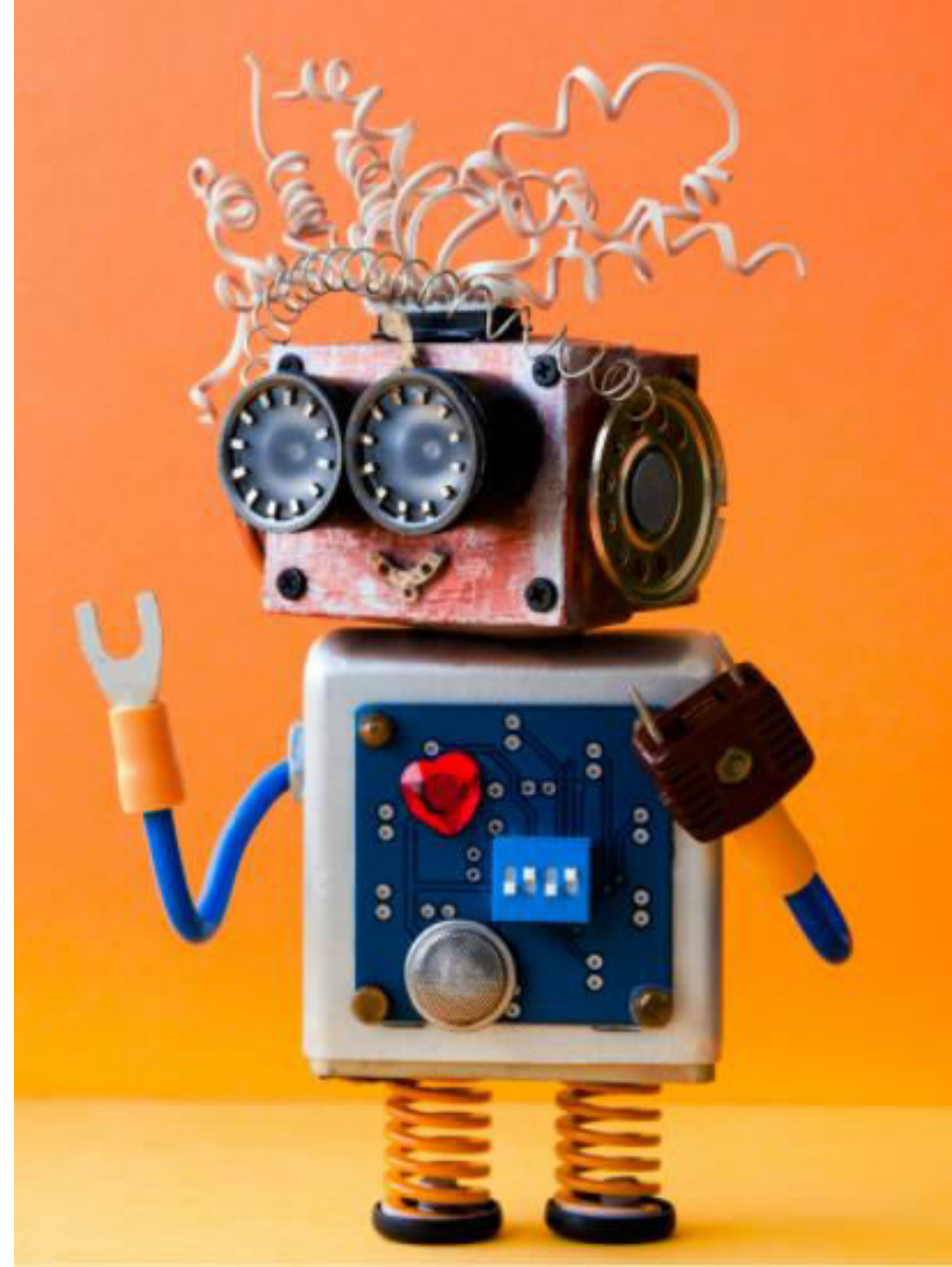
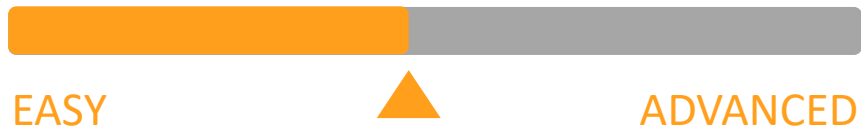
- All you need to do is to use these two lines of code!
- Please note that hyperparameters and all other model training aspects are set to default.

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



Source: https://auto.gluon.ai/stable/tutorials/tabular_prediction/tabular-quickstart.html

AUTOGLUON PRESETS AND FIT METHOD PARAMETERS



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.
 - 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

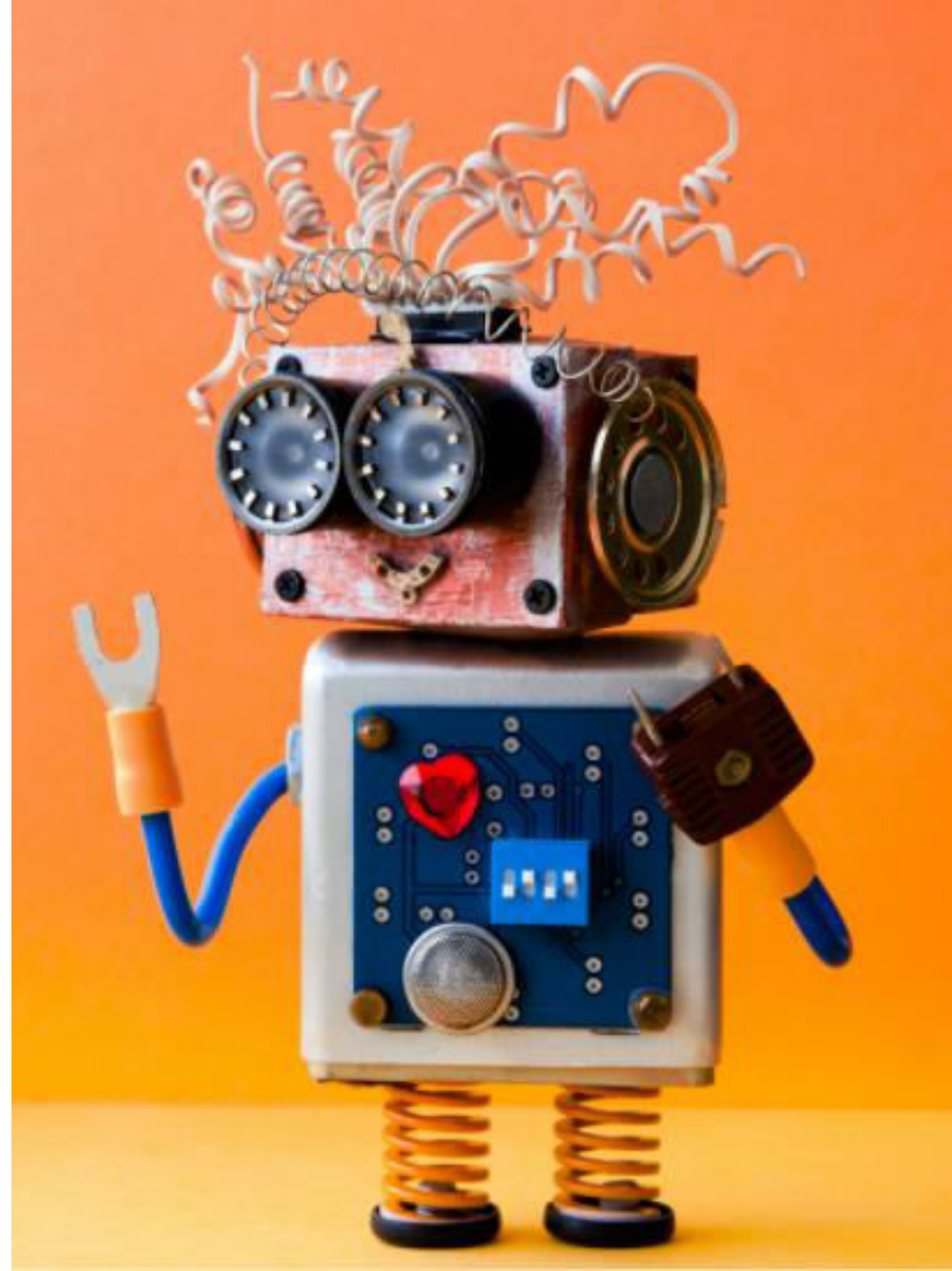
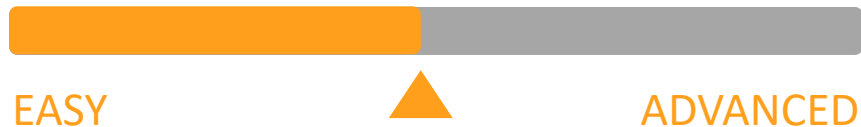
```
fit(train_data, tuning_data=None, time_limit=None, presets=None, hyperparameters=None, feature_metadata='infer',  
    **kwargs) [source]
```

- 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, and inference latency of trained models
 - 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']

AUTOGLUON 101: PRESETS

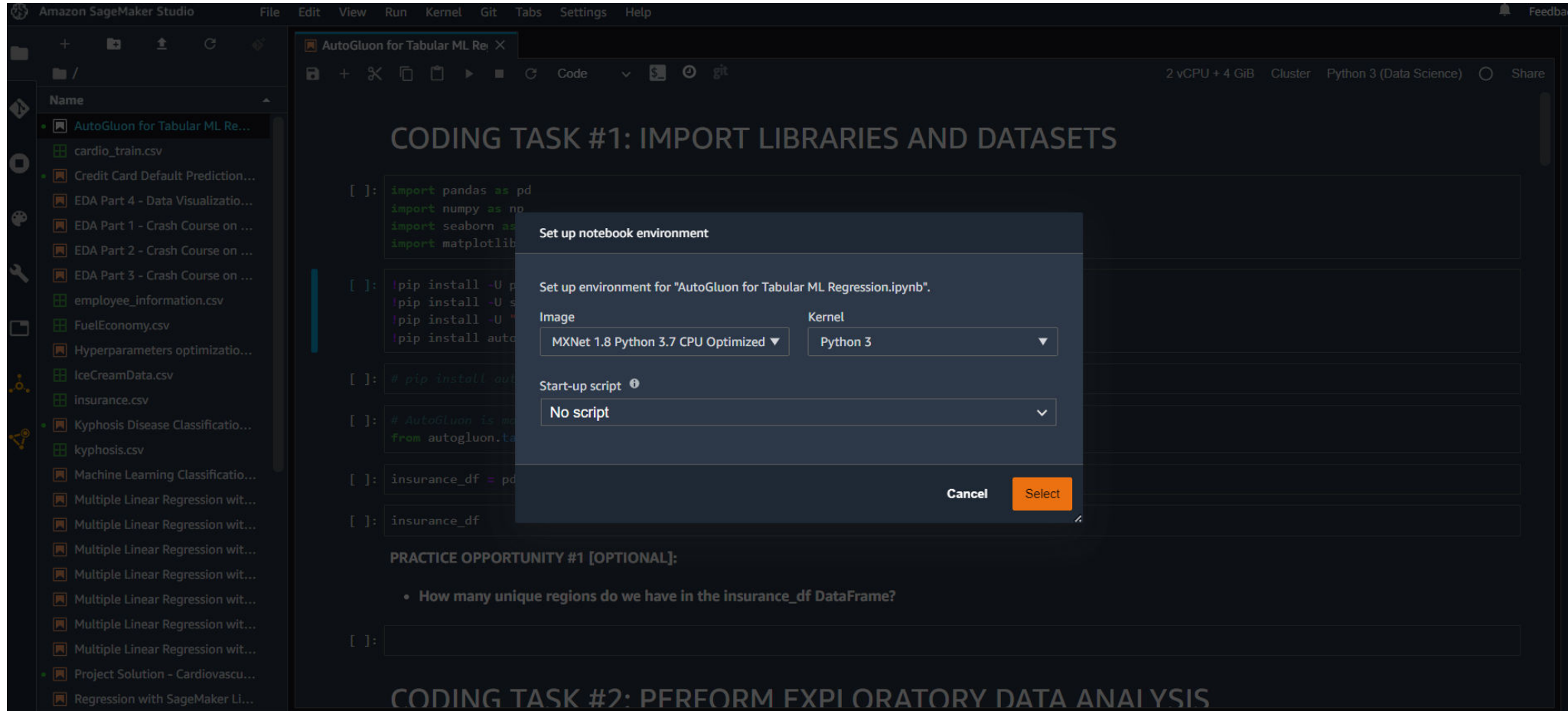
- **best_quality:** Best predictive accuracy with little consideration to inference time or disk usage.
- **high_quality_fast_inference_only_refit:** High predictive accuracy with fast inference. ~10x-200x faster inference and ~10x-200x lower disk usage than best_quality. Recommended for applications 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: REGRESSION MODELS TRAINING USING AUTOGLUON



AUTOGLUON FOR REGRESSION 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')

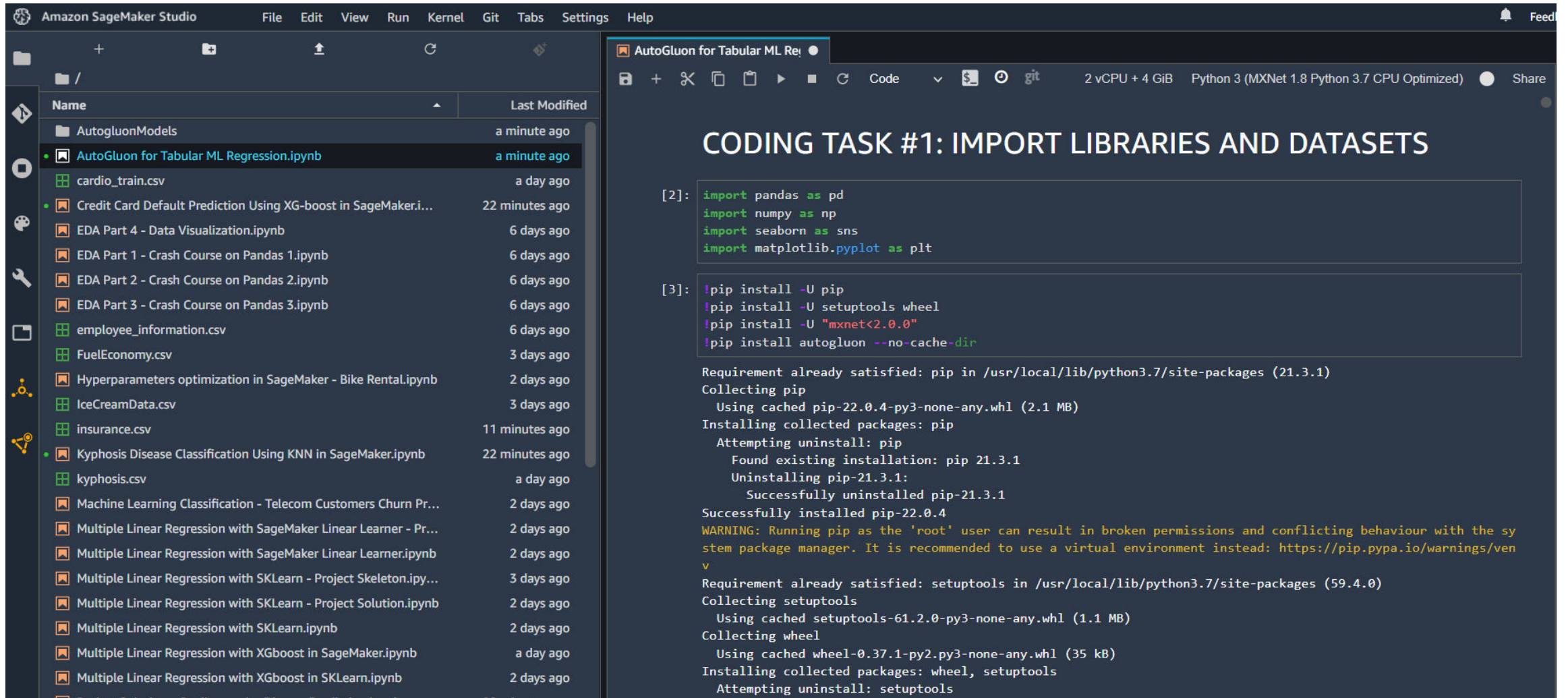
[ ]: insurance_df
```

A modal dialog titled 'Set up notebook environment' is open, prompting the user to configure the 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 question: 'How many unique regions do we have in the insurance_df DataFrame?'. The notebook also displays 'CODING TASK #1: IMPORT LIBRARIES AND DATASETS' and 'CODING TASK #2: PERFORM EXPLORATORY DATA ANALYSIS'.

AUTOGLUON FOR REGRESSION DEMO



The screenshot displays the Amazon SageMaker Studio interface. On the left, a file explorer shows a list of files and notebooks, including 'AutogluonModels', 'AutoGluon for Tabular ML Regression.ipynb', 'cardio_train.csv', 'Credit Card Default Prediction Using XG-boost in SageMaker.i...', 'EDA Part 4 - Data Visualization.ipynb', 'EDA Part 1 - Crash Course on Pandas 1.ipynb', 'EDA Part 2 - Crash Course on Pandas 2.ipynb', 'EDA Part 3 - Crash Course on Pandas 3.ipynb', 'employee_information.csv', 'FuelEconomy.csv', 'Hyperparameters optimization in SageMaker - Bike Rental.ipynb', 'IceCreamData.csv', 'insurance.csv', 'Kyphosis Disease Classification Using KNN in SageMaker.ipynb', 'kyphosis.csv', 'Machine Learning Classification - Telecom Customers Churn Pr...', 'Multiple Linear Regression with SageMaker Linear Learner - Pr...', 'Multiple Linear Regression with SageMaker Linear Learner.ipynb', 'Multiple Linear Regression with SKLearn - Project Skeleton.ipy...', 'Multiple Linear Regression with SKLearn - Project Solution.ipynb', 'Multiple Linear Regression with SKLearn.ipynb', 'Multiple Linear Regression with XGboost in SageMaker.ipynb', and 'Multiple Linear Regression with XGboost in SKLearn.ipynb'.

The main area shows a Jupyter notebook titled 'AutoGluon for Tabular ML Regression.ipynb'. The notebook contains the following code:

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

[3]: !pip install -U pip
!pip install -U setuptools wheel
!pip install -U "mxnet<2.0.0"
!pip install autogluon --no-cache-dir
```

The output of the code shows the following messages:

```
Requirement already satisfied: pip in /usr/local/lib/python3.7/site-packages (21.3.1)
Collecting pip
Using cached pip-22.0.4-py3-none-any.whl (2.1 MB)
Installing collected packages: pip
Attempting uninstall: pip
Found existing installation: pip 21.3.1
Uninstalling pip-21.3.1:
Successfully uninstalled pip-21.3.1
Successfully installed pip-22.0.4
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the sy
stem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/ven
v
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/site-packages (59.4.0)
Collecting setuptools
Using cached setuptools-61.2.0-py3-none-any.whl (1.1 MB)
Collecting wheel
Using cached wheel-0.37.1-py2.py3-none-any.whl (35 kB)
Installing collected packages: wheel, setuptools
Attempting uninstall: setuptools
```


AUTOGLUON FOR REGRESSION DEMO

TRAINING MODELS USING AUTOGLUON IS VERY
STRAIGHTFORWARD! 1-2 LINES OF CODE AND YOU'RE ALL SET!

TASK #4: TRAIN MULTIPLE MODELS USING AUTOGLUON

```
[*]: # Split the data into 80% for training and 20% for testing using train_test_split
from sklearn.model_selection import train_test_split
X_train, X_test = train_test_split(insurance_df, test_size=0.2, random_state=0)
```

```
[*]: X_train
```

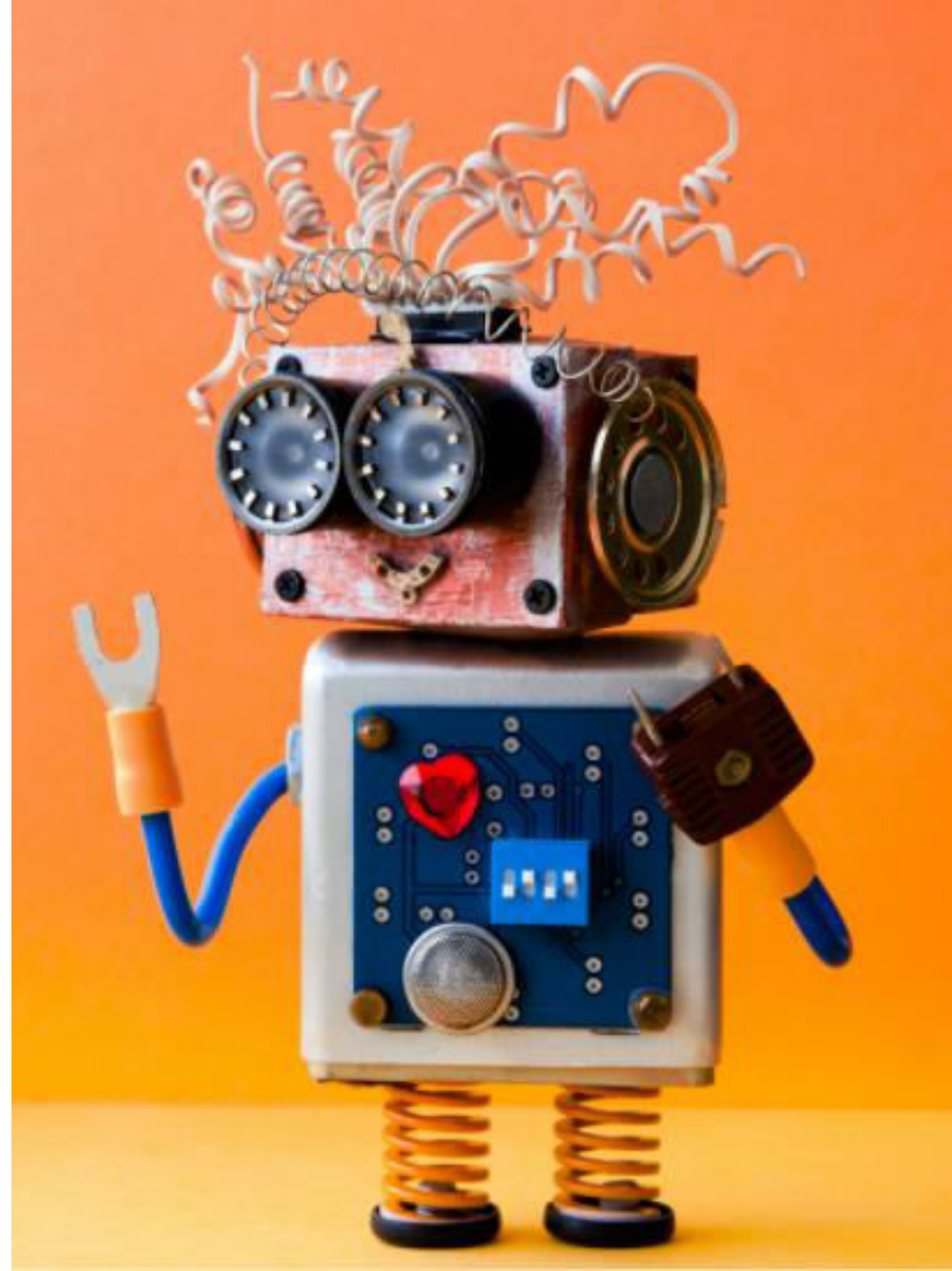
```
[*]: X_test
```

```
[*]: # 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
# For regression type problems, 'label' values are generally floating point non-integers with large number of unique values

predictor = TabularPredictor(label="charges", problem_type = 'regression', eval_metric = 'r2').fit(train_data = X_train, time_limit = 200, presets = "best_quality")
```

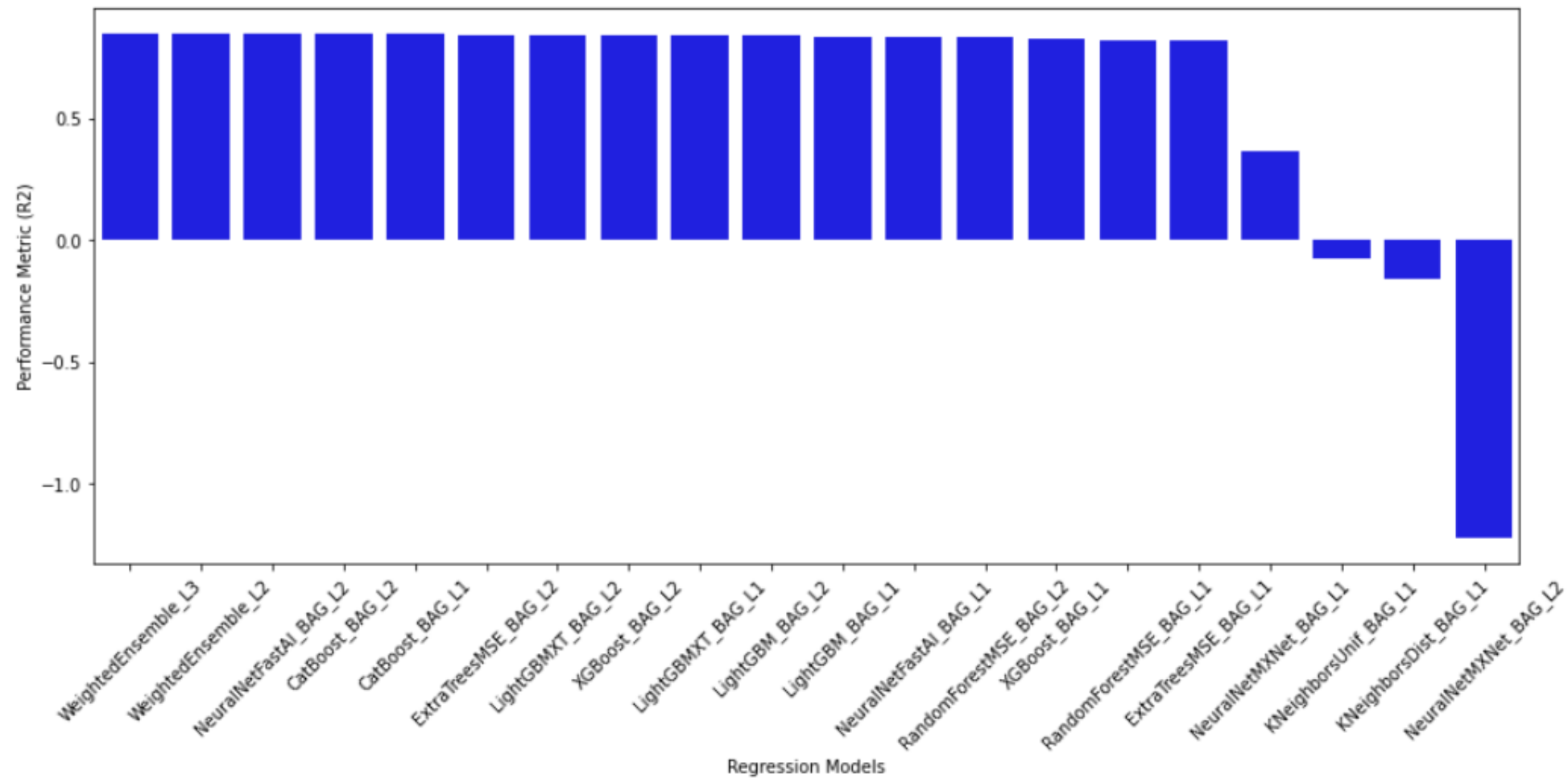
```
[*]: predictor.fit_summary()
```

DEMO PART #2: REGRESSION MODELS EVALUATION USING AUTOGLUON



AUTOGLUON FOR REGRESSION DEMO

REGRESSION MODELS LEADERBOARD



AUTOGLUON FOR REGRESSION DEMO

	model	score_val	pred_time_val	fit_time	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer	fit_order
0	WeightedEnsemble_L3	0.852081	2.881016	155.410457	0.000482	0.370429	3	True	20
1	WeightedEnsemble_L2	0.848804	2.024641	110.830163	0.000491	0.348534	2	True	11
2	NeuralNetFastAI_BAG_L2	0.848142	2.654688	132.482453	0.185246	11.665695	2	True	17
3	CatBoost_BAG_L2	0.847701	2.507353	132.038746	0.037912	11.221988	2	True	15
4	CatBoost_BAG_L1	0.846395	0.035744	4.540622	0.035744	4.540622	1	True	6
5	ExtraTreesMSE_BAG_L2	0.845681	2.588426	121.760457	0.118985	0.943700	2	True	16
6	LightGBMXT_BAG_L2	0.843113	2.521876	127.245895	0.052434	6.429137	2	True	12
7	XGBoost_BAG_L2	0.842892	2.538392	131.208645	0.068950	10.391887	2	True	18
8	LightGBMXT_BAG_L1	0.842331	0.051749	5.789112	0.051749	5.789112	1	True	3
9	LightGBM_BAG_L2	0.838843	2.520250	128.188247	0.050808	7.371490	2	True	13
10	LightGBM_BAG_L1	0.838494	0.049149	5.089594	0.049149	5.089594	1	True	4
11	NeuralNetFastAI_BAG_L1	0.836701	0.144982	11.828727	0.144982	11.828727	1	True	8
12	RandomForestMSE_BAG_L2	0.834860	2.589095	122.955369	0.119654	2.138611	2	True	14
13	XGBoost_BAG_L1	0.831363	0.059599	3.908278	0.059599	3.908278	1	True	9
14	RandomForestMSE_BAG_L1	0.822675	0.115800	0.843566	0.115800	0.843566	1	True	5
15	ExtraTreesMSE_BAG_L1	0.820145	0.118055	0.627691	0.118055	0.627691	1	True	7
16	NeuralNetMXNet_BAG_L1	0.367723	1.678475	88.179121	1.678475	88.179121	1	True	10
17	KNeighborsUnif_BAG_L1	-0.072961	0.103590	0.005625	0.103590	0.005625	1	True	1
18	KNeighborsDist_BAG_L1	-0.159388	0.112299	0.004422	0.112299	0.004422	1	True	2
19	NeuralNetMXNet_BAG_L2	-1.222989	4.069572	134.266658	1.600130	13.449900	2	True	19

AUTOGLUON FOR REGRESSION DEMO

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

```
Evaluation: r2 on test data: 0.9020321906625249
```

```
Evaluations on test data:
```

```
{  
  "r2": 0.9020321906625249,  
  "root_mean_squared_error": -3948.368499602538,  
  "mean_squared_error": -15589613.808653597,  
  "mean_absolute_error": -2427.890058888526,  
  "pearsonr": 0.9503944589237935,  
  "median_absolute_error": -1697.0432498046875  
}
```

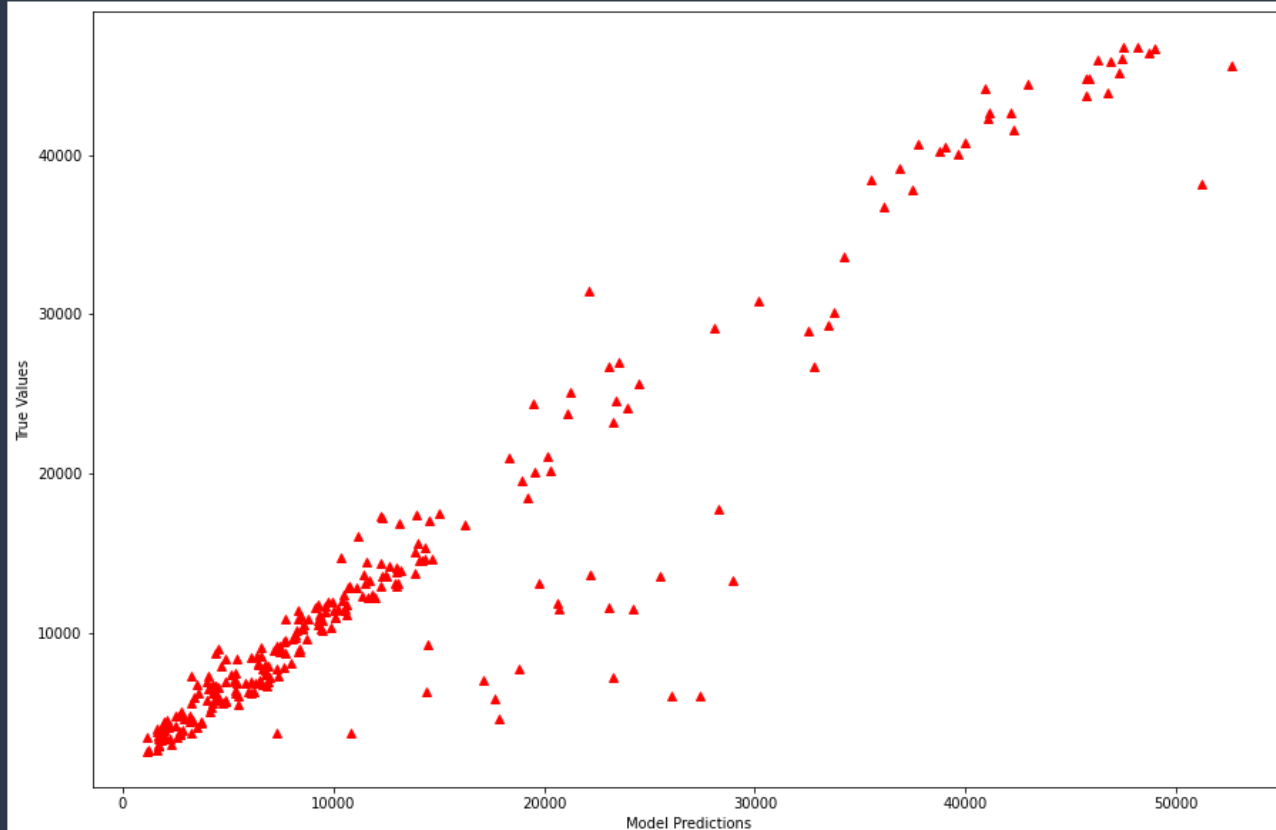
```
[26]: {'r2': 0.9020321906625249,  
      'root_mean_squared_error': -3948.368499602538,  
      'mean_squared_error': -15589613.808653597,  
      'mean_absolute_error': -2427.890058888526,  
      'pearsonr': 0.9503944589237935,  
      'median_absolute_error': -1697.0432498046875}
```

AUTOGLUON FOR REGRESSION DEMO

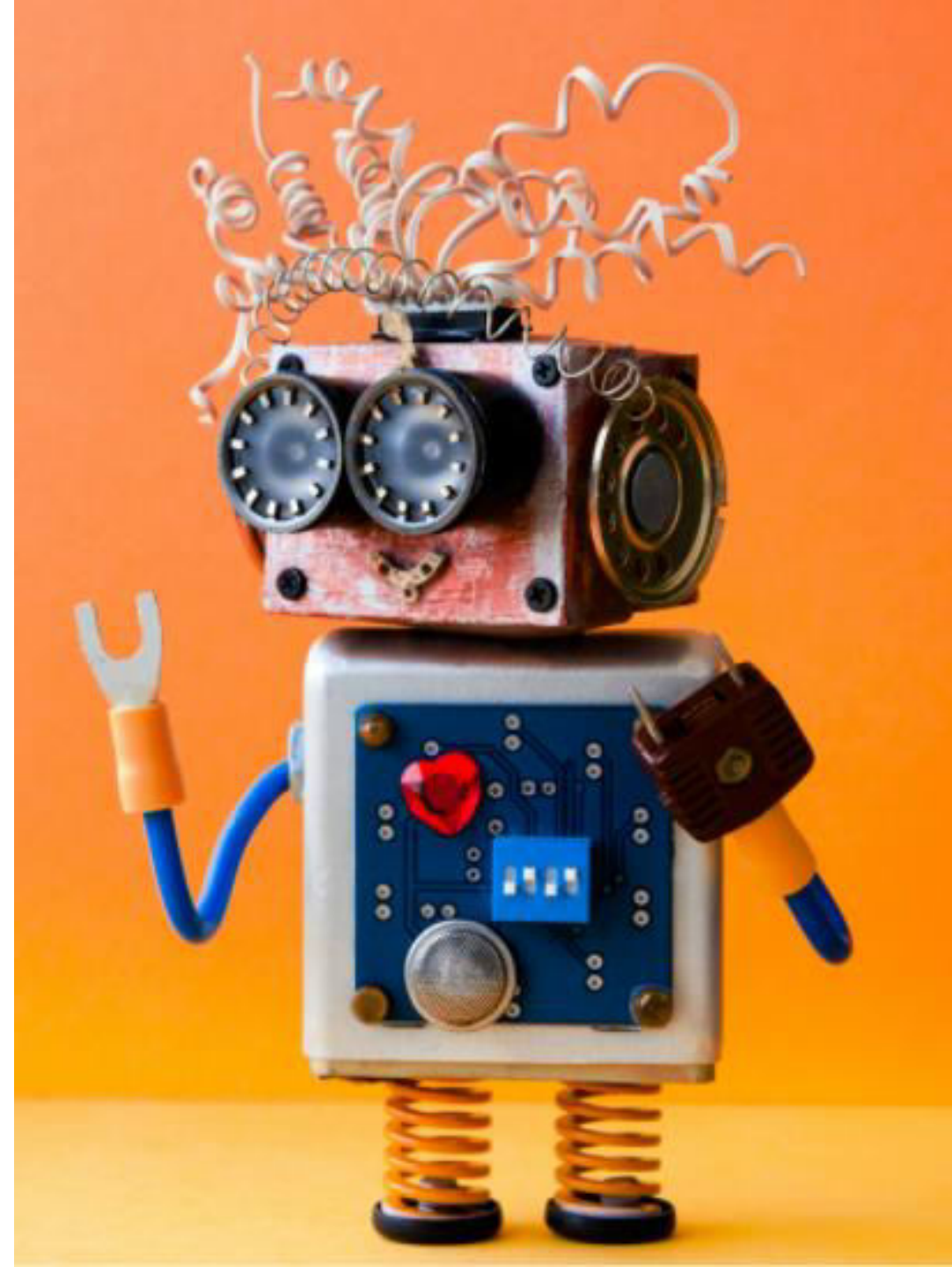
PLOT MODEL PREDICTIONS VS. TRUE VALUES (GROUNDTRUTH FROM TESTING DATASETS)

```
[32]: y_predict = predictor.predict(X_test)
plt.figure(figsize = (15, 10))
plt.plot(y_test, y_predict, "^", color = 'r')
plt.xlabel('Model Predictions')
plt.ylabel('True Values')
```

```
[32]: Text(0, 0.5, 'True Values')
```



FINAL END-OF-DAY CAPSTONE PROJECT



PROJECT OVERVIEW

- *The goal of this project is to use Autogluon to build, train, and test several machine learning models to predict bike rental usage using inputs such as temperature, humidity, wind speed..etc.*
- *This project can be effectively used by bike rental shops to predict demand and expected future sales and understand key factors that contribute to generating revenue.*



Image Source: <https://pixabay.com/photos/bike-rental-bike-rental-photos-pasa/6757993805>

Dataset Source: Hadi Fanaee-T, Laboratory of Artificial Intelligence and Decision Support (LIAAD), University of Porto INESC Porto, Campus da FEUP Rua Dr. Roberto Frias, 378 4200 - 465 Porto, Portugal
<https://www.kaggle.com/mjafajugazyan/cars1>

PROJECT OVERVIEW: INPUTS AND OUTPUTS

- **Inputs:**

- instant: record index
- dteday: date
- season: season (1: springer, 2: summer, 3: fall, 4: winter)
- yr: year (0: 2011, 1: 2012)
- mnth: month (1 to 12)
- hr: hour (0 to 23)
- holiday: whether day is holiday or not - weekday : day of the week
- Working day: if day is neither weekend nor holiday is 1, otherwise is 0.
- weathersit :
 - ❖ 1: Clear, Few clouds, Partly cloudy
 - ❖ 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
 - ❖ 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
 - ❖ 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp : Normalized temperature in Celsius. The values are divided to 41 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)

- **Outputs:**

- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

MODEL OVERVIEW

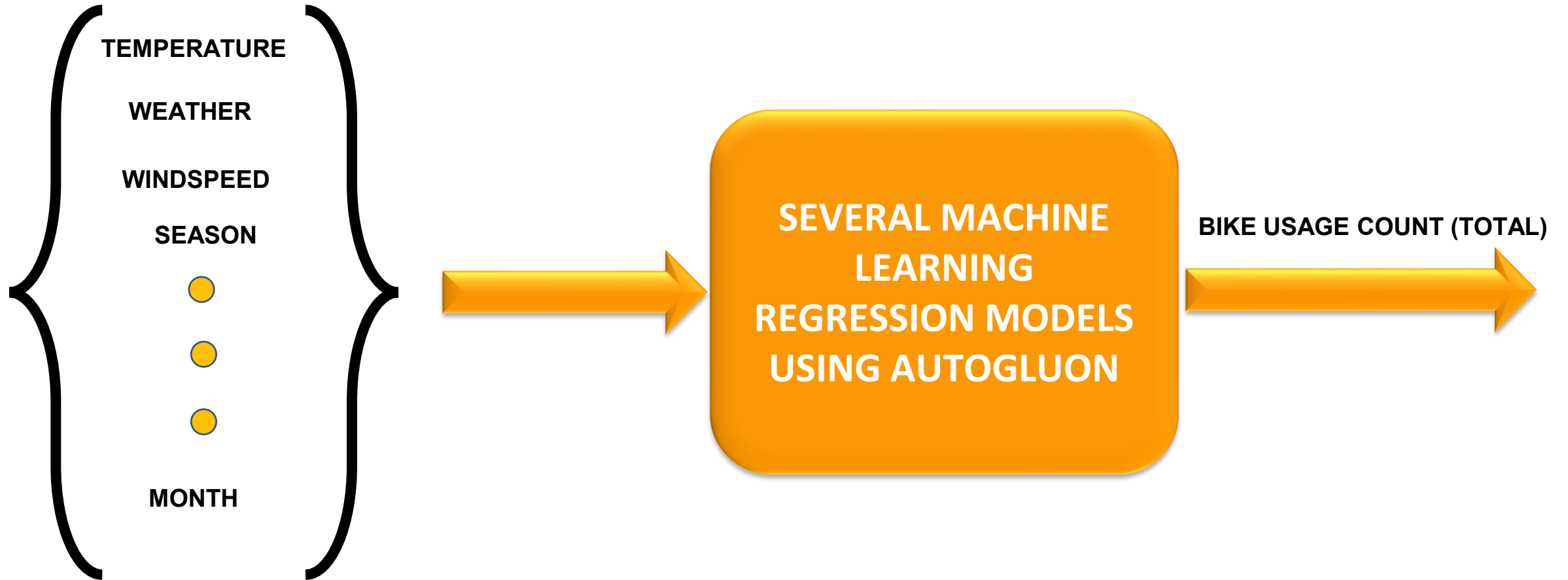


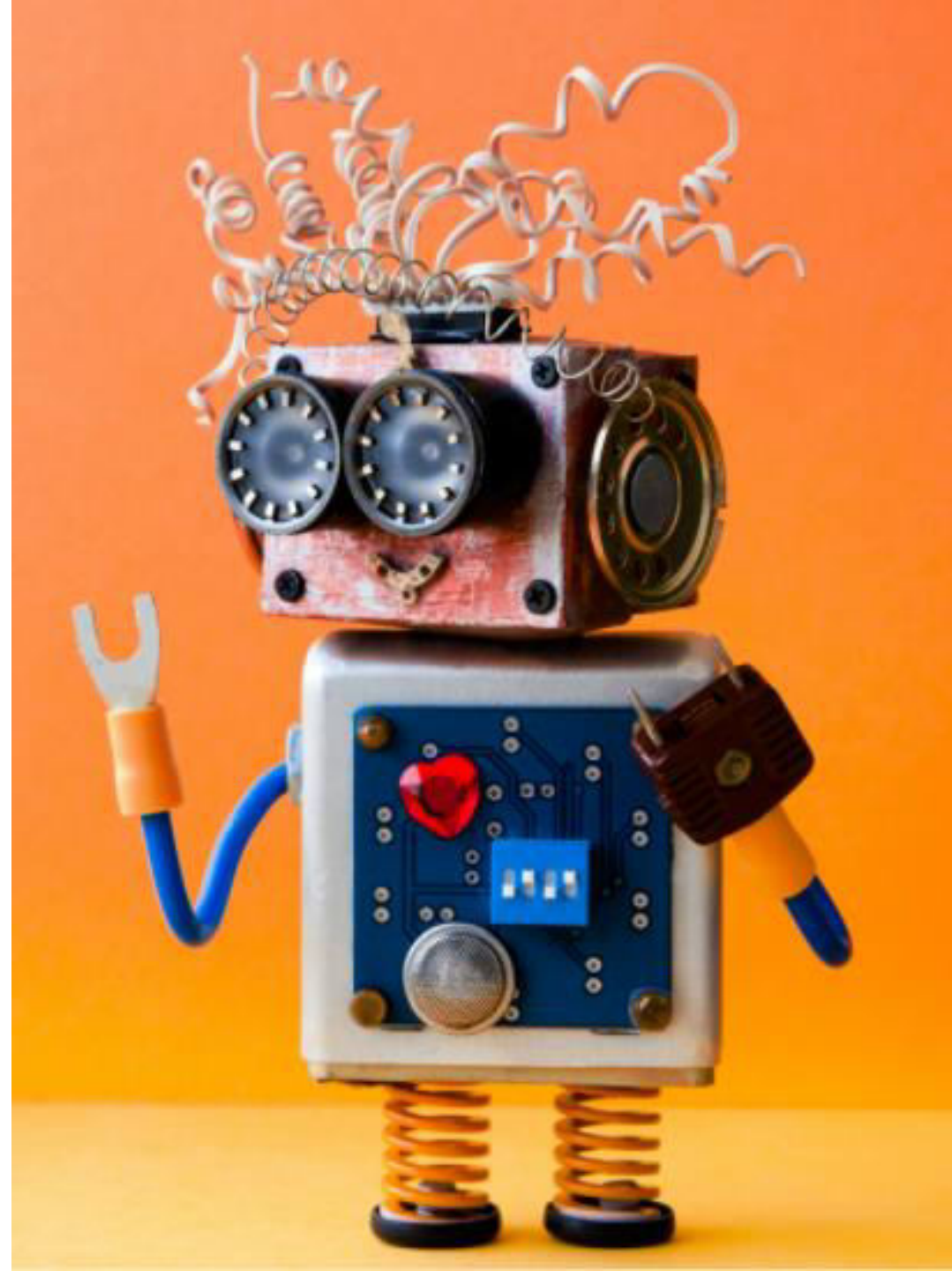
Photo Credit: https://commons.wikimedia.org/wiki/File:Neural_network.svg

PROJECT TASKS

Please complete the following:

1. Load the “*bike_sharing_daily.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 R2 metric, train machine linear regression models using AutoGluon to predict the “casual” column only (Note: ignore total count and registered)
5. Assess trained models' performance by plotting the leaderboard and indicate the best model
6. Using ‘optimized for deployment’ preset and RMSE metric, train machine linear regression models using AutoGluon to predict the “casual” column only (Note: ignore total count and registered)
7. Assess trained models' performance by plotting the leaderboard and indicate the best model

FINAL END-OF-DAY CAPSTONE SOLUTION



PROJECT SOLUTION

```
[45]: predictor.leaderboard()
```

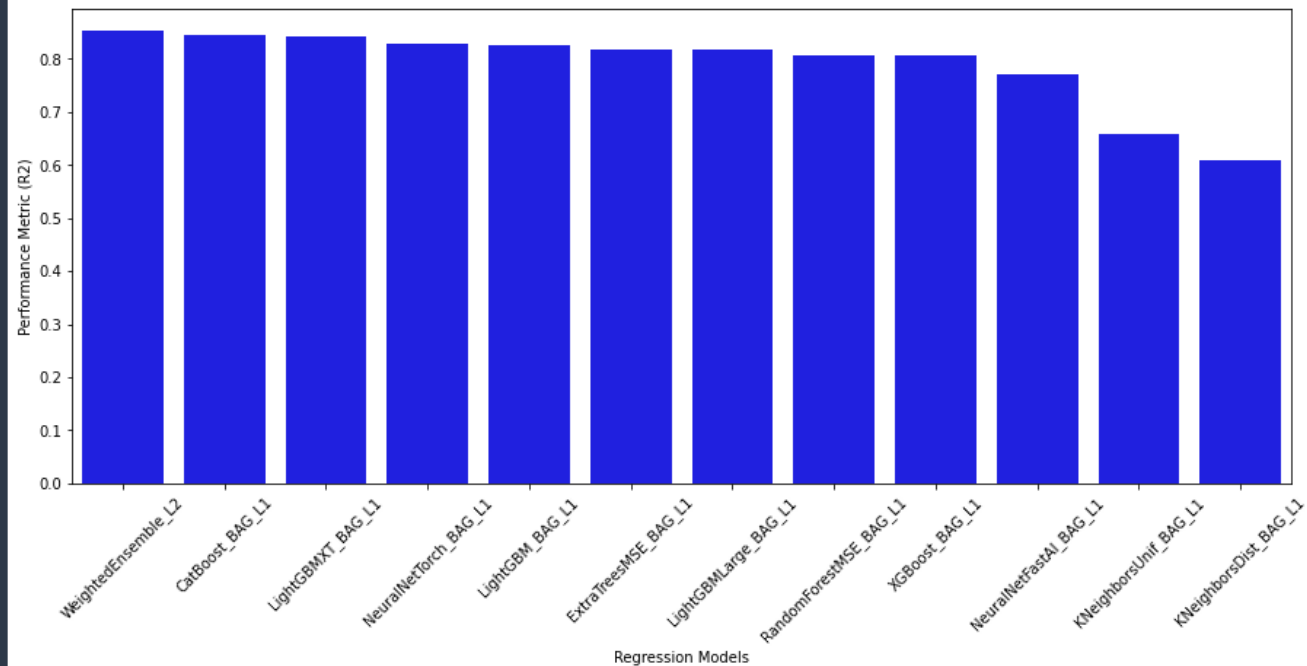
	model	score_val	pred_time_val	fit_time	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer	fit_order
0	WeightedEnsemble_L2	0.851747	0.169931	58.685316	0.000703	0.646562	2	True	12
1	CatBoost_BAG_L1	0.844732	0.012148	16.015900	0.012148	16.015900	1	True	6
2	LightGBMXT_BAG_L1	0.842214	0.064328	11.866491	0.064328	11.866491	1	True	3
3	NeuralNetTorch_BAG_L1	0.827885	0.047695	18.301743	0.047695	18.301743	1	True	10
4	LightGBM_BAG_L1	0.824676	0.037627	11.138051	0.037627	11.138051	1	True	4
5	ExtraTreesMSE_BAG_L1	0.817766	0.129039	0.805564	0.129039	0.805564	1	True	7
6	LightGBMLarge_BAG_L1	0.816326	0.119066	19.726948	0.119066	19.726948	1	True	11
7	RandomForestMSE_BAG_L1	0.805007	0.124551	1.137770	0.124551	1.137770	1	True	5
8	XGBoost_BAG_L1	0.804843	0.045057	11.854619	0.045057	11.854619	1	True	9
9	NeuralNetFastAI_BAG_L1	0.770147	0.094161	16.469966	0.094161	16.469966	1	True	8
10	KNeighborsUnif_BAG_L1	0.656510	0.102633	0.011768	0.102633	0.011768	1	True	1
11	KNeighborsDist_BAG_L1	0.609321	0.101758	0.006698	0.101758	0.006698	1	True	2

```
[45]:
```

	model	score_val	pred_time_val	fit_time	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer	fit_order
0	WeightedEnsemble_L2	0.851747	0.169931	58.685316	0.000703	0.646562	2	True	12
1	CatBoost_BAG_L1	0.844732	0.012148	16.015900	0.012148	16.015900	1	True	6
2	LightGBMXT_BAG_L1	0.842214	0.064328	11.866491	0.064328	11.866491	1	True	3
3	NeuralNetTorch_BAG_L1	0.827885	0.047695	18.301743	0.047695	18.301743	1	True	10
4	LightGBM_BAG_L1	0.824676	0.037627	11.138051	0.037627	11.138051	1	True	4
5	ExtraTreesMSE_BAG_L1	0.817766	0.129039	0.805564	0.129039	0.805564	1	True	7
6	LightGBMLarge_BAG_L1	0.816326	0.119066	19.726948	0.119066	19.726948	1	True	11
7	RandomForestMSE_BAG_L1	0.805007	0.124551	1.137770	0.124551	1.137770	1	True	5
8	XGBoost_BAG_L1	0.804843	0.045057	11.854619	0.045057	11.854619	1	True	9
9	NeuralNetFastAI_BAG_L1	0.770147	0.094161	16.469966	0.094161	16.469966	1	True	8
10	KNeighborsUnif_BAG_L1	0.656510	0.102633	0.011768	0.102633	0.011768	1	True	1
11	KNeighborsDist_BAG_L1	0.609321	0.101758	0.006698	0.101758	0.006698	1	True	2

PROJECT SOLUTION

	model	score_val	pred_time_val	fit_time	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer	fit_order
0	WeightedEnsemble_L2	0.851747	0.169931	58.685316	0.000703	0.646562	2	True	12
1	CatBoost_BAG_L1	0.844732	0.012148	16.015900	0.012148	16.015900	1	True	6
2	LightGBMXt_BAG_L1	0.842214	0.064328	11.866491	0.064328	11.866491	1	True	3
3	NeuralNetTorch_BAG_L1	0.827885	0.047695	18.301743	0.047695	18.301743	1	True	10
4	LightGBM_BAG_L1	0.824676	0.037627	11.138051	0.037627	11.138051	1	True	4
5	ExtraTreesMSE_BAG_L1	0.817766	0.129039	0.805564	0.129039	0.805564	1	True	7
6	LightGBMLarge_BAG_L1	0.816326	0.119066	19.726948	0.119066	19.726948	1	True	11
7	RandomForestMSE_BAG_L1	0.805007	0.124551	1.137770	0.124551	1.137770	1	True	5
8	XGBoost_BAG_L1	0.804843	0.045057	11.854619	0.045057	11.854619	1	True	9
9	NeuralNetFastAI_BAG_L1	0.770147	0.094161	16.469966	0.094161	16.469966	1	True	8
10	KNeighborsUnif_BAG_L1	0.656510	0.102633	0.011768	0.102633	0.011768	1	True	1
11	KNeighborsDist_BAG_L1	0.609321	0.101758	0.006698	0.101758	0.006698	1	True	2



PROJECT SOLUTION

```
Evaluation: r2 on test data: 0.8797139356963792
```

```
Evaluations on test data:
```

```
{  
  "r2": 0.8797139356963792,  
  "root_mean_squared_error": -252.10995271120268,  
  "mean_squared_error": -63559.42825604485,  
  "mean_absolute_error": -165.6636921889117,  
  "pearsonr": 0.9411193208366362,  
  "median_absolute_error": -104.973388671875  
}
```


PROJECT SOLUTION

