A popular algorithm for clustering data is the Adaptive Resonance Theory (ART) family of algorithms—a set of neural network models that you can use for pattern recognition and prediction. The ART1 algorithm maps an input vector to a neuron in a recognition field through a weight vector. A key advantage of ART is that it incorporates a vigilance parameter to determine how closely a feature vector must map to a cluster for it to be a member. Otherwise, it can create a new cluster automatically for the feature vector. The vigilance parameter dictates whether there are many clusters that more closely match the input vectors (high vigilance) or general clusters that incorporate more feature vector examples (low vigilance).

Another neural network method, called self-organizing map (SOM), translates multidimensional feature vectors into a low-dimensional representation called a map. This method implements a form of data compression through vector quantization. The neural network makes up the map, which is typically a regular lattice shape (NxM neurons representing a rectangular grid). Each neuron in the map has a weight vector that matches the dimensionality of the input vector

**Create a machine learning web app to predict your insurance premium cost**

## Summary

As we see the value of gross insurance premiums worldwide continue to skyrocket past 5 trillion dollars, we know that most of these costs are preventable. For example, just by eliminating smoking and lowering your BMI by a few points might mean shaving thousands of dollars from your premium charges. In this application, we study the effects of age, smoking, BMI, gender, and region to determine how much of a difference these factors can make on your insurance premium. By using our application, customers see the radical difference their lifestyle choices make on their insurance charges. By leveraging artificial intelligence (AI) and machine learning, we help customers understand just how much smoking increases their premium by predicting how much they will have to pay within seconds.

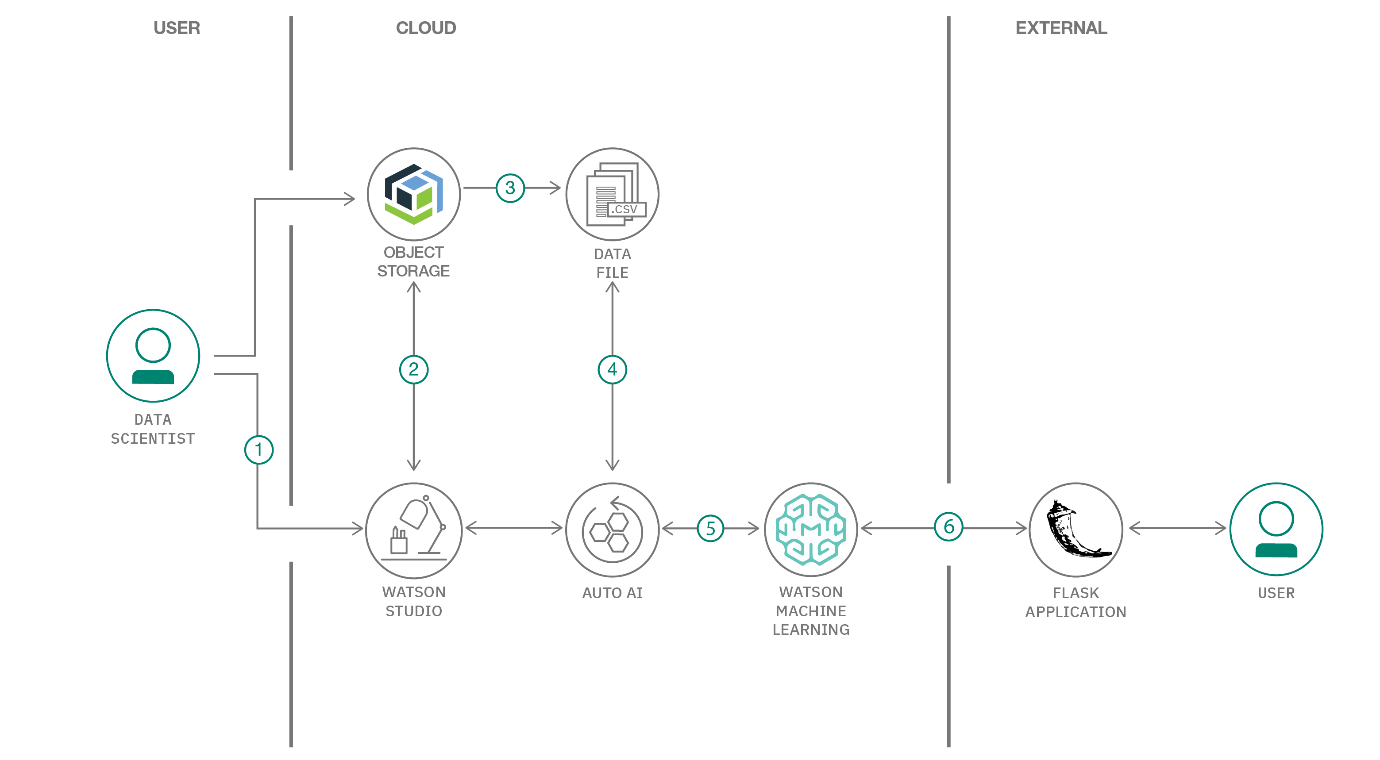
## Description

Using IBM AutoAI, we automate all of the tasks involved in building predictive models for different requirements. You see how AutoAI generates great models quickly, which saves time and effort, and aids in a faster decision-making process. You create a model from a data set that includes the age, gender, BMI, number of children, smoking preferences, region, and charges to predict the health insurance premium cost that an individual pays.

When you have completed this code pattern, you understand how to:

* Quickly set up the services on IBM Cloud to build the model
* Ingest the data and initiate the AutoAI process
* Build different models using AutoAI and evaluate the performance
* Choose the best model and complete the deployment
* Generate predictions using the deployed model by making REST calls
* Compare the process of using AutoAI and building the model manually
* Visualize the deployed model using a front-end application

## Flow



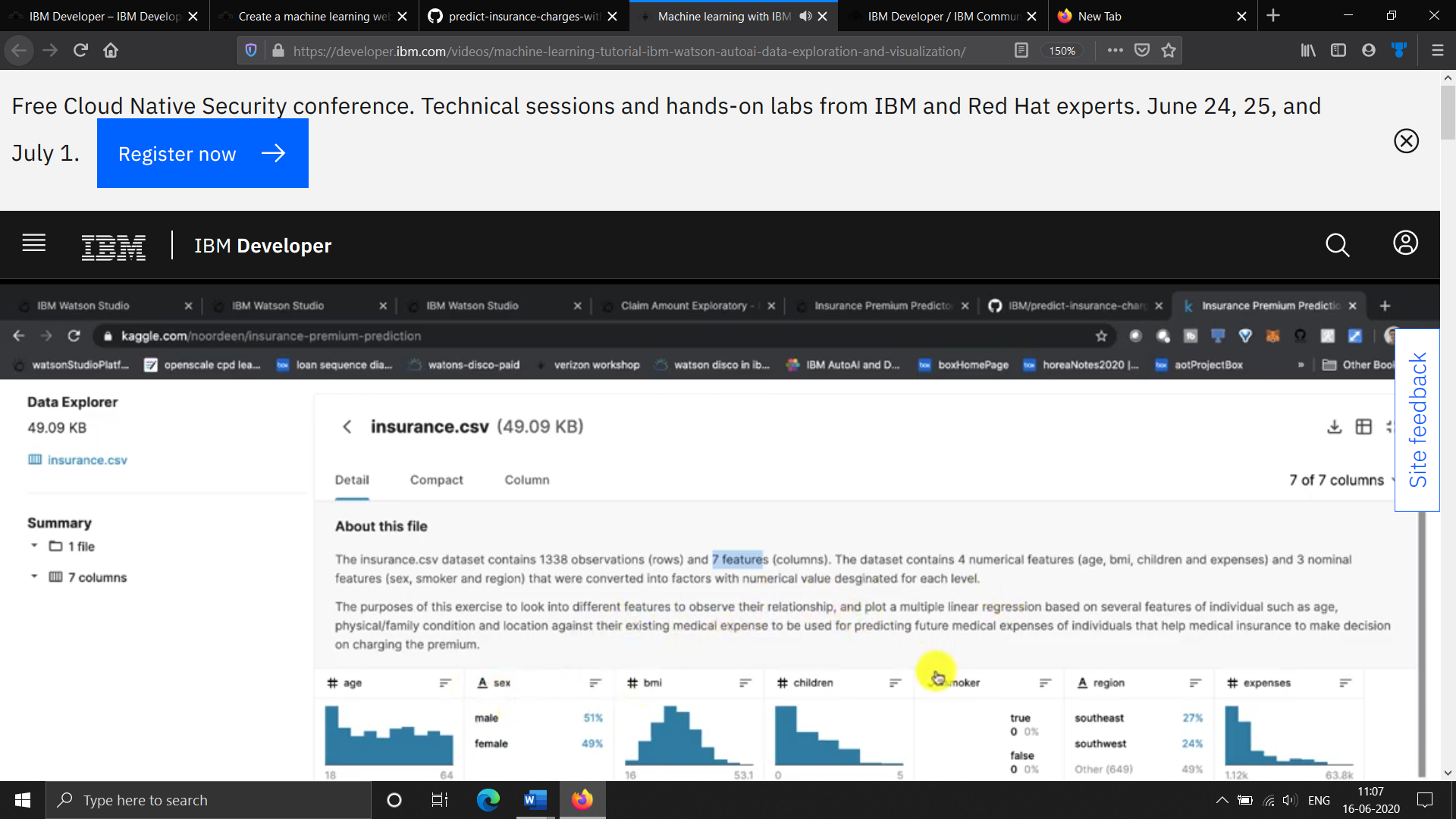
1. The user creates an IBM Watson Studio Service on IBM Cloud.
2. The user creates an IBM Cloud Object Storage Service and adds that to Watson Studio.
3. The user uploads the insurance premium data file into Watson Studio.
4. The user creates an AutoAI Experiment to predict an insurance premium on Watson Studio.
5. AutoAI uses Watson Machine Learning to create several models, and the user deploys the best performing model.
6. The user uses the Flask web application to connect to the deployed model and predict an insurance charge.

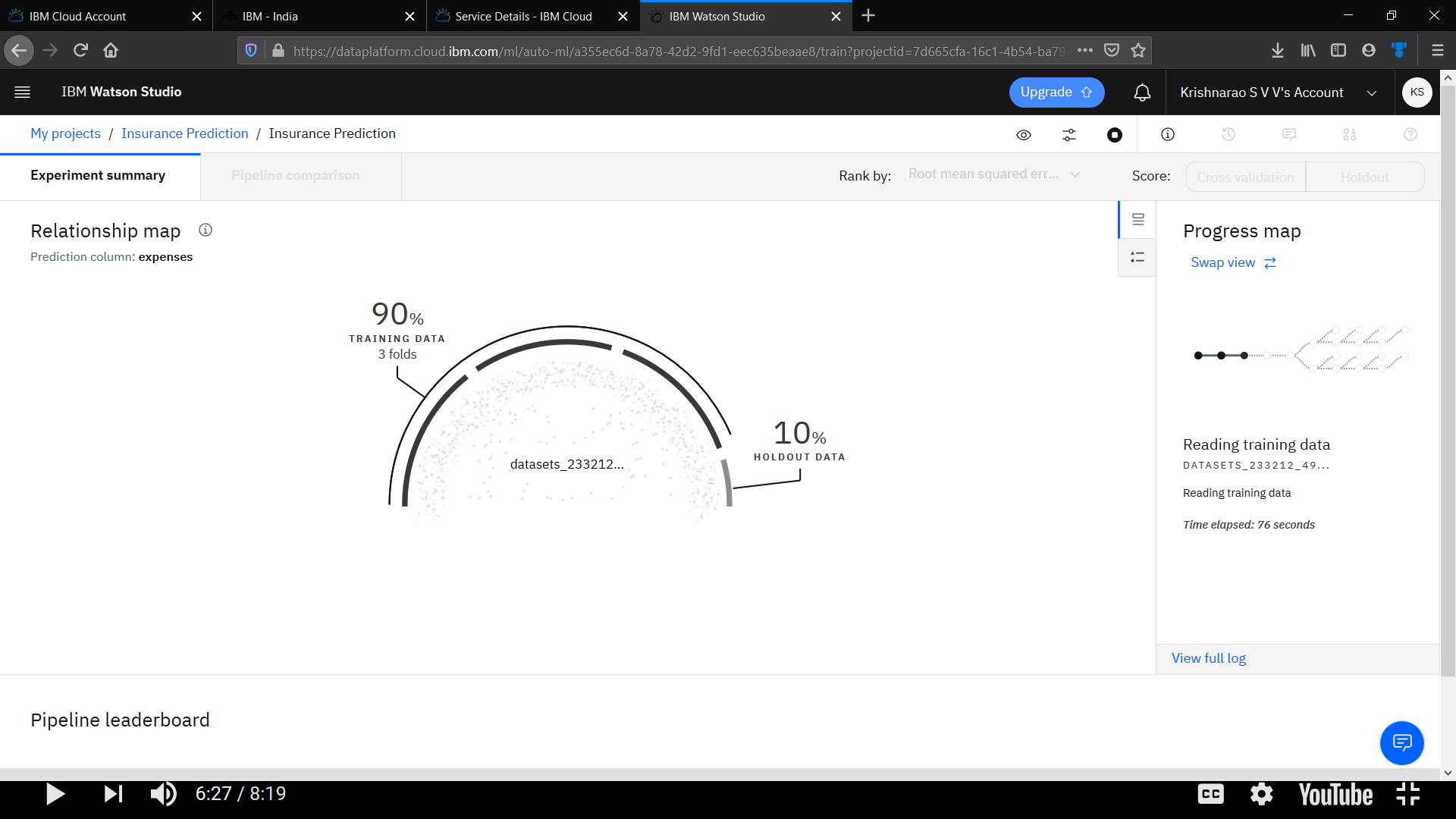
## Instructions

Get detailed instructions in the [readme](https://github.com/IBM/predict-insurance-charges-with-ai/blob/master/README.md) file. Those steps tell you how to:

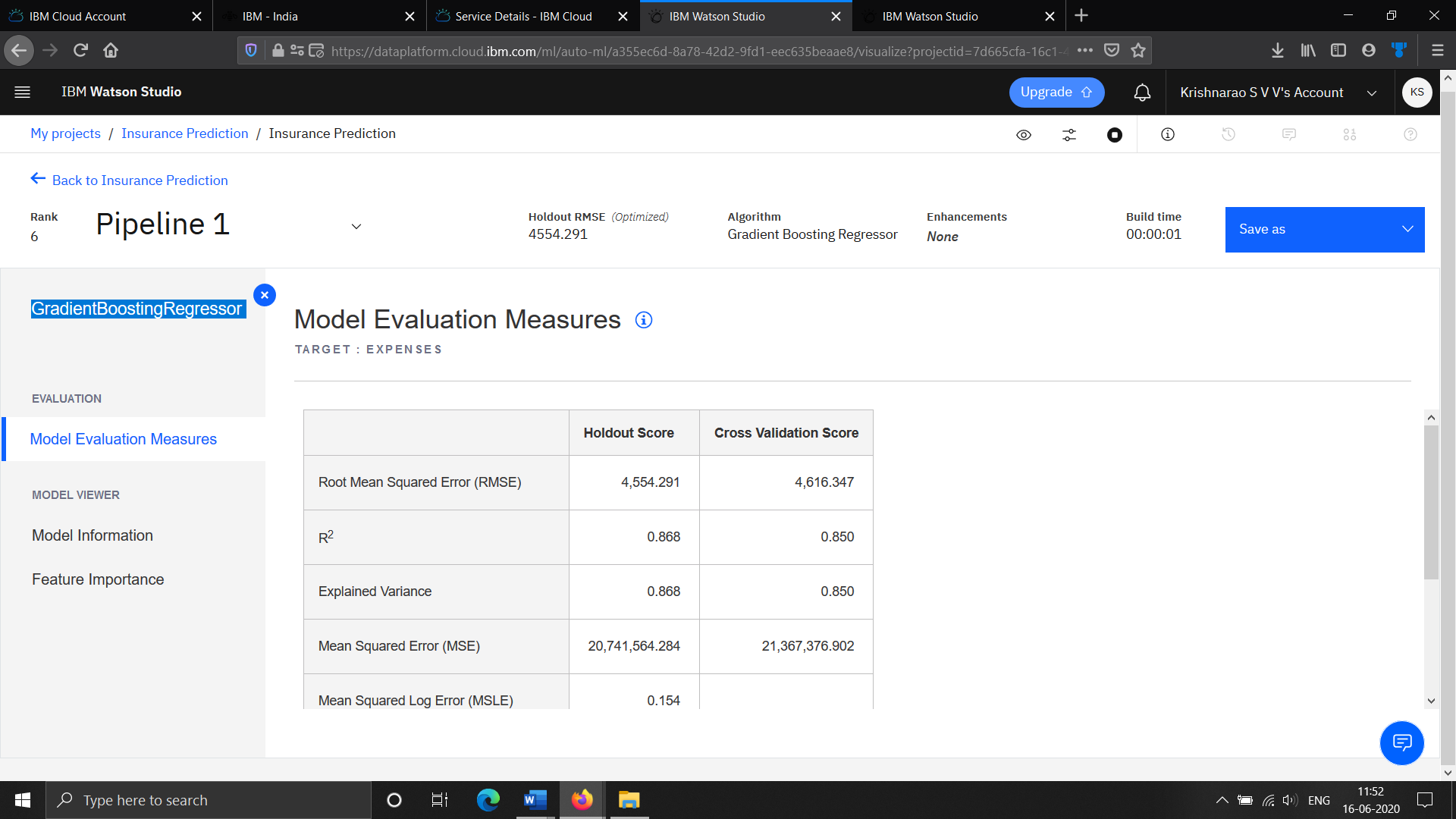
1. Download the data set.
2. Clone the repo.
3. Explore the data (optional).
4. Create IBM Cloud services.
5. Create and run AutoAI Experiment.
6. Create a deployment and test your model.
7. Create a notebook from your model (optional).
8. Run the application.

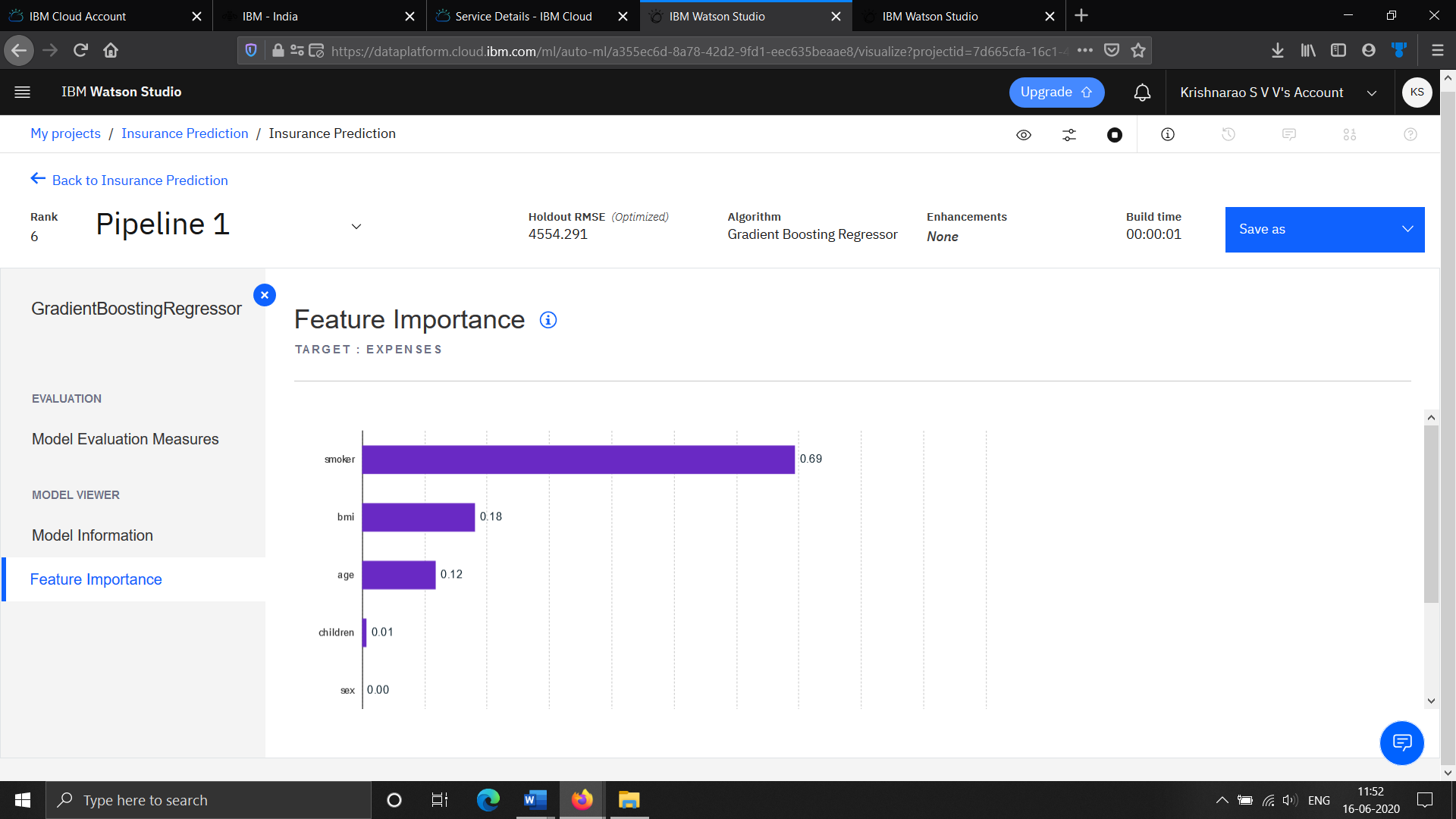
Github page : IBM/predict-insurance-charges-with-ai





GradientBoostingRegressor





Model Information

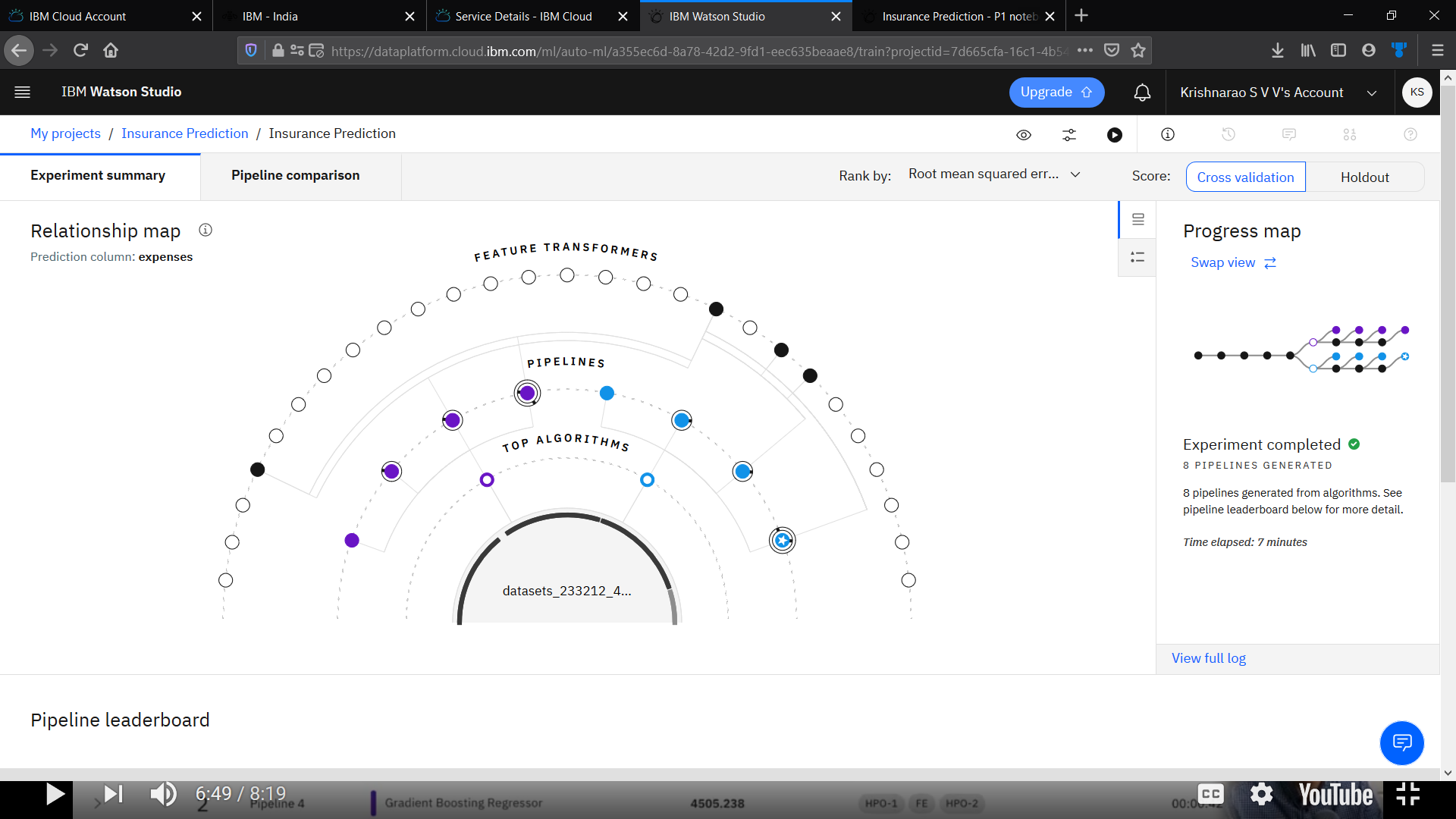
Target : expenses

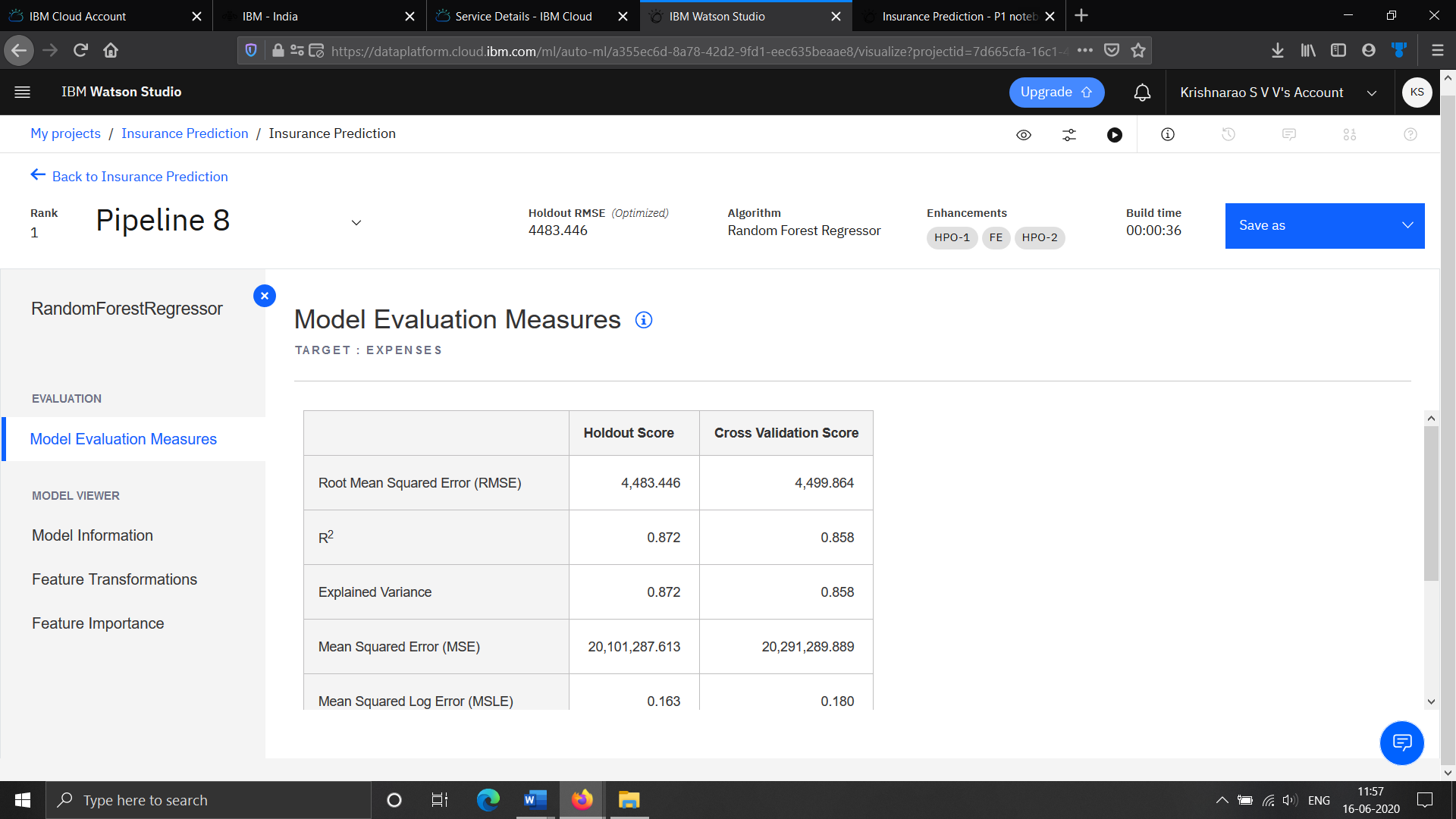
|  |  |
| --- | --- |
| **Label (Target)** | expenses |
| **Number of Features** | 6 |
| **Created At** | 6/16/2020, 11:47:50 AM |

Model Evaluation Measures

Target : expenses

|  | **Holdout Score** | **Cross Validation Score** |
| --- | --- | --- |
| **Root Mean Squared Error (RMSE)** | 4,554.291 | 4,616.347 |
| **R2** | 0.868 | 0.850 |
| **Explained Variance** | 0.868 | 0.850 |
| **Mean Squared Error (MSE)** | 20,741,564.284 | 21,367,376.902 |
| **Mean Squared Log Error (MSLE)** | 0.154 |  |
| **Mean Absolute Error (MAE)** | 2,441.955 | 2569.6723528392182,569.672 |





**Predictive modeling using tools such as SAS, Minitab, SPSS**

## Algorithms used for classification models

These algorithms are the default algorithms used for automatic model selection for classification problems.

| **Algorithm** | **Description** |
| --- | --- |
| Decision Tree Classifier | Maps observations about an item (represented in branches) to conclusions about the item’s target value (represented in leaves). Supports both binary and multiclass labels, as well as both continuous and categorical features. |
| Extra Trees Classifier | An averaging algorithm based on randomized decision trees. |
| Gradient Boosted Tree Classifier | Produces a classification prediction model in the form of an ensemble of decision trees. It only supports binary labels, as well as both continuous and categorical features. |
| LGBM Classifier | Gradient boosting framework that uses leaf-wise (horizontal) tree-based learning algorithm. |
| Logistic Regression | Analyzes a data set in which there are one or more independent variables that determine one of two outcomes. Only binary logistic regression is supported |
| Random Forest Classifier | Constructs multiple decision trees to produce the label that is a mode of each decision tree. It supports both binary and multiclass labels, as well as both continuous and categorical features. |
| XGBoost Classifier | Accurate sure procedure that can be used for classification problems. XGBoost models are used in a variety of areas including Web search ranking and ecology. |

## Algorithms used for regression models

These algorithms are the default algorithms used for automatic model selection for regression problems.

| **Algorithm** | **Description** |
| --- | --- |
| Decision Tree Regression | Maps observations about an item (represented in the branches) to conclusions about the item’s target value (represented in the leaves). It supports both continuous and categorical features. |
| Extra Trees Regression | An averaging algorithm based on randomized decision trees. |
| Gradient Boosting Regression | Produces a regression prediction model in the form of an ensemble of decision trees. It supports both continuous and categorical features. |
| LGBM Regression | Gradient boosting framework that uses tree-based learning algorithms. |
| Linear Regression | Models the linear relationship between a scalar-dependent variable y and one or more explanatory variables (or independent variables) x. |
| Random Forest Regression | Constructs multiple decision trees to produce the mean prediction of each decision tree. It supports both continuous and categorical features. |
| Ridge | Ridge regression is similar to Ordinary Least Squares but imposes a penalty on the size of coefficients. |
| XGBoost Regression | GBRT is an accurate and effective off-the-shelf procedure that can be used for regression problems. Gradient Tree Boosting models are used in a variety of areas including Web search ranking and ecology. |

## Metrics by model type

The following metrics are available for measuring the accuracy of pipelines during training and when scoring data.

### Binary classification metrics

* Roc auc (default for ranking the pipelines)
* Accuracy
* Average precision
* F
* Negative log loss
* Precision
* Recall

### Multi-class classification metrics

Metrics for multi-class models can be adjusted to account for imbalances in labels. for example:

* Metrics with the micro qualifier calculate metrics globally by counting the total true positives, false negatives and false positives.
* Metrics with the micro qualifier calculates metrics for each label, and finds their unweighted mean. This does not take label imbalance into account.
* Metrics with the weighted qualifier calculate metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters macro to account for label imbalance; it can result in an F-score that is not between precision and recall.

These are the multi-class classification metrics:

* Accuracy (default for ranking the pipelines)
* F1
* F1 Micro
* F1 Macro
* F1 Weighted
* Precision
* Precision Micro
* Precision Macro
* Precision Weighted
* Recall
* Recall Micro
* Recall Macro
* Recall Weighted

### Regression metrics

* Negative root mean squared error (default for ranking the pipeline)
* Negative mean absolute error
* Negative root mean squared log error
* Explained variance
* Negative mean squared error
* Negative mean squared log error
* Negative median absolute error
* R2

### Metrics used for feature importance

Feature importance is calculated from the average of nine measures applied to the training data:

* Linear Correlation (f\_regression) metric
* Maximal Information Coefficient (MIC) metric
* Linear Regression (LR) metric
* L1 regularization metric (Lasso)
* Ridge metric
* RF metric
* Stability Selection
* Recursive Feature Elimination (RFE)
* Recursive Feature Elimination plus selection of best number of features

## Data transformations

For feature engineering, AutoAI uses a novel approach that explores various feature construction choices in a structured, non-exhaustive manner, while progressively maximizing model accuracy using reinforcement learning. This results in an optimized sequence of transformations for the data that best match the algortihms, or estimators, of the model selection step. This table lists some of the transformations used and some well-known conditions under which they are useful. This is not an exhaustive list of scenarios where the transformation is useful, as that can be complex and hard to interpret. Finally, the listed scenarios are not an explanation of how the transformations are selected. The selection of which transforms to apply is done in a trial and error, performance-oriented manner.

| **Name** | **Code** | **Function** |
| --- | --- | --- |
| Principle Component Analysis | pca | Reduce dimensions of data and realign across a more suitable coordinate system. Helps tackle the ‘curse of dimensionality’ in linearly correlated data. It eliminates redundancy and separates significant signals in data. |
| Standard Scaler | stdscaler | Scales data features to a standard range.This helps the efficacy and efficiency of certain learning algorithms as well as other transformations such as PCA. |
| Logarithm | log | Reduces right skewness in features and make them more symmetric. Resulting symmetry in features helps algorithms understand the data better. Even scaling based on mean and variance is more meaningful on symmetrical data. Additionally, it can capture specific physical relationships between feature and target best described through a logarithm. |
| Cube Root | cbrt | Reduces right skewness in data like logarithm, but is weaker than log in its impact, which might be more suitable in some cases. It is also applicable to negative or zero values to which log doesn’t apply. Cube root can also change units such as reducing volume to length. |
| Square root | sqrt | Reduces mild right skewness in data. It is weaker than log or cube root. It works with zeroes and reduces spatial dimensions such as area to length. |
| Square | square | Reduces left skewness to a moderate extent to make such distributions more symmetric.It can also be helpful in capturing certain phenomena such as super-linear growth. |
| Product | product | A product of two features can expose a non-linear relationship to better predict the target value than the individual values alone. For example, item cost into number of items sold is a better indication of the size of a business than any of those alone. |
| Numerical XOR | nxor | This transform helps capture “exclusive disjunction” type of relationships between variables, similar to a bitwise XOR, but in a general numerical context. |
| Sum | sum | Sometimes the sum of two features is better correlated to the prediction target than the features alone. For instance, loans from different sources, when summed up, provide a better idea of a credit applicant’s total indebtedness. |
| Divide | divide | Division is a fundamental operand used to express quantities such as gross GDP over population (per capita GDP), representing a country’s average lifespan better than either GDP alone or population alone. |
| Maximum | max | Take the higher of two values. |
| Rounding | round | This transformation can be seen as perturbation or adding some noise to reduce overfitting that might have been a result of inaccurate observations. |
| Absolute Value | abs | Consider only the magnitude and not the sign of observation. Sometimes, the direction or sign of an observation doesn’t matter so much as the magnitude of it, such as physical displacement, while considering fuel or time spent in the actual movement. |
| Hyperbolic tangent | tanh | Non-linear activation function can improve prediction accuracy, similar to that of neural network activation functions. |
| Sine | sin | Can reorient data to discover periodic trends such as simple harmonic motions. |
| Cosine | cos | Can reorient data to discover periodic trends such as simple harmonic motions. |
| Tangent | tan | Trigonometric tangent transform is usually helpful in combination with other transforms. |
| Feature Agglomeration | featureagglomeration | Clustering different features into groups, based upon distance or affinity, provides ease of classification for the learning algorithm. |
| Sigmoid | sigmoid | Non-linear activation function can improve prediction accuracy, similar to that of neural network activation functions. |

## AutoAI FAQs

The following are commonly asked questions about creating an AutoAI experiment.

### How many pipelines are created?

Two AutoAI parameters determine the number of pipelines:

* **max\_num\_daub\_ensembles:** Maximum number (top-K ranked by DAUB model selection) of the selected algorithm, or estimator types, for example LGBMClassifierEstimator, XGBoostClassifierEstimator, or LogisticRegressionEstimator to use in pipeline composition.  The default is 1, where only the highest ranked by model selection algorithm type is used.
* **num\_folds:** Number of subsets of the full dataset to train pipelines in addition to the full dataset. The default is 1 for training the full data set.           For each fold and algorithm type, AutoAI creates 4 pipelines of increased refinement, corresponding to:

1. Pipeline with default sklearn parameters for this algorithm type,
2. Pipeline with optimized algorithm using HPO
3. Pipeline with optimized feature engineering
4. Pipeline with optimized feature engineering and optimized algorithm using HPO

The total number of pipelines generated is :

TotalPipelines= max\_num\_daub\_ensembles \* 4, if num\_folds = 1:

TotalPipelines= (num\_folds+1) \* max\_num\_daub\_ensembles \* 4,  if num\_folds > 1 :

### What hyperparameter optimization is applied to my model?

AutoAI uses a model-based, derivative-free global search algorithm, called RBfOpt, which is tailored for the costly machine learning model training and scoring evaluations required by hyperparameter optimization (HPO). In contrast to Bayesian optimization, which fits a Gaussian model to the unknown objective function, RBfOpt fits a radial basis function mode to accelerate the discovery of hyper-parameter configurations that maximize the objective function of the machine learning problem at hand. This acceleration is achieved by minimizing the number of expensive training and scoring machine learning models evaluations and by eliminating the need to compute partial derivatives.

For each fold and algorithm type, AutoAI creates two pipelines that use HPO to optimize for the algorithm type.

* The first is based on optimizing this algorithm type based on the preprocessed (imputed/encoded/scaled) dataset (pipeline 2) above).
* The second is based on optimizing the algorithm type based on optimized feature engineering of the preprocessed (imputed/encoded/scaled) data set.

The parameter values of the algorithms of all pipelines generated by AutoAI is published in status messages.

For more details regarding the RbfOpt algorithm, see:

* [RbfOpt: A blackbox optimization library in Python](https://github.com/coin-or/rbfopt)
* [An effective algorithm for hyperparameter optimization of neural networks. IBM Journal of Research and Development, 61(4-5), 2017](http://ieeexplore.ieee.org/document/8030298/)

curl -k -X POST `

>> --header "Content-Type: application/x-www-form-urlencoded" `

>> --header "Accept: application/json" `

>> --data-urlencode "grant\_type=urn:ibm:params:oauth:grant-type:apikey" `

>> --data-urlencode "apikey=RODj2WbUXBZrJQh\_SGtcGTjyAvd3dg7JNgNLAbyrx5Oe" `

>> <https://iam.bluemix.net/identity/token>

curl -X POST `

--header "Content-Type: application/x-www-form-urlencoded" `

--header "Accept: application/json" `

--data-urlencode "grant\_type=urn:ibm:params:oauth:grant-type:apikey" `

--data-urlencode "apikey= " `

<https://iam.bluemix.net/identity/token>

curl -X POST `

--header "Content-Type: application/x-www-form-urlencoded" `

--header "Accept: application/json" `

--data-urlencode "grant\_type=urn:ibm:params:oauth:grant-type:apikey" `

--data-urlencode "apikey= " `

"https://iam.bluemix.net/identity/token"

curl -X POST \

'https://iam.cloud.ibm.com/identity/token' \

-H 'Content-Type: application/x-www-form-urlencoded' \

-d 'grant\_type=urn:ibm:params:oauth:grant-type:apikey&apikey=MY\_APIKEY'

curl -k -X POST \

--header "Content-Type: application/x-www-form-urlencoded" \

--header "Accept: application/json" \

--data-urlencode "grant\_type=urn:ibm:params:oauth:grant-type:apikey" \

--data-urlencode "apikey= " \

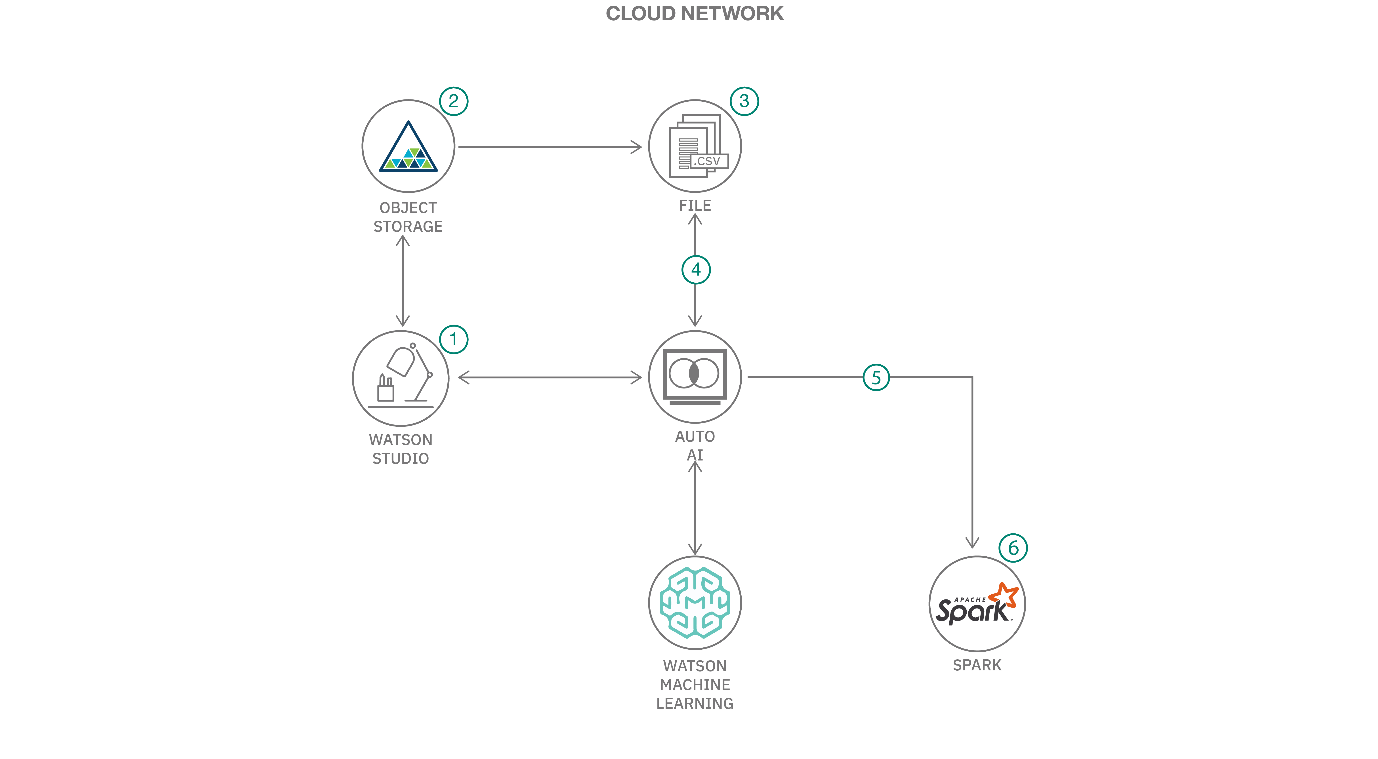
"https://iam.bluemix.net/identity/token"

AutoAI can be run in public clouds and in private clouds, including IBM Cloud Pak™ for Data.

When you have completed this code pattern, you understand how to:

* Quickly set up the model building services on IBM Cloud
* Ingest the data and initiate the AutoAI tool
* Build different models using AutoAI and evaluate the performance
* Choose the best model and complete the deployment
* Generate predictions using the deployed model by making REST calls
* Compare the process of using AutoAI and building the model manually

**Flow**

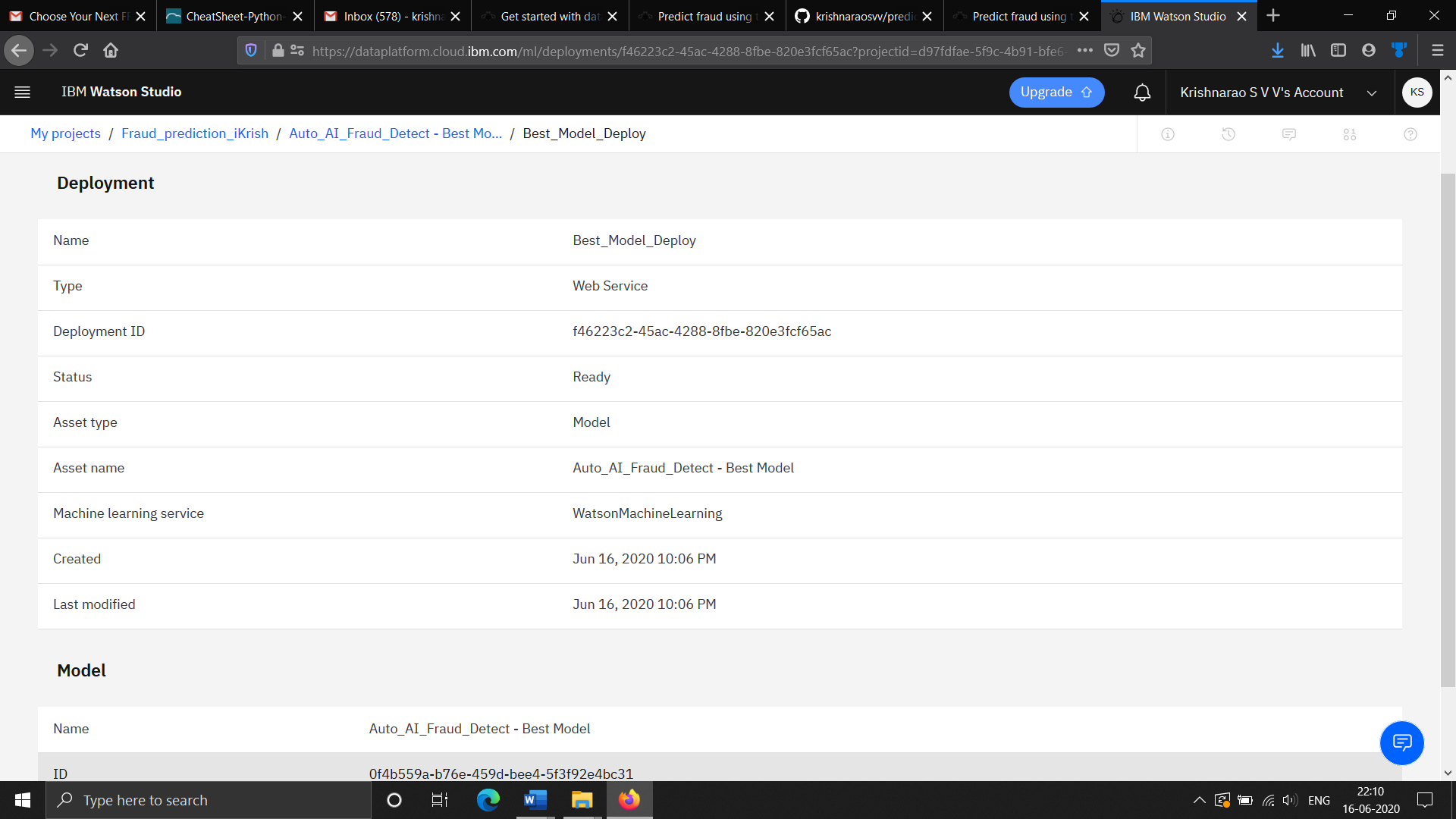


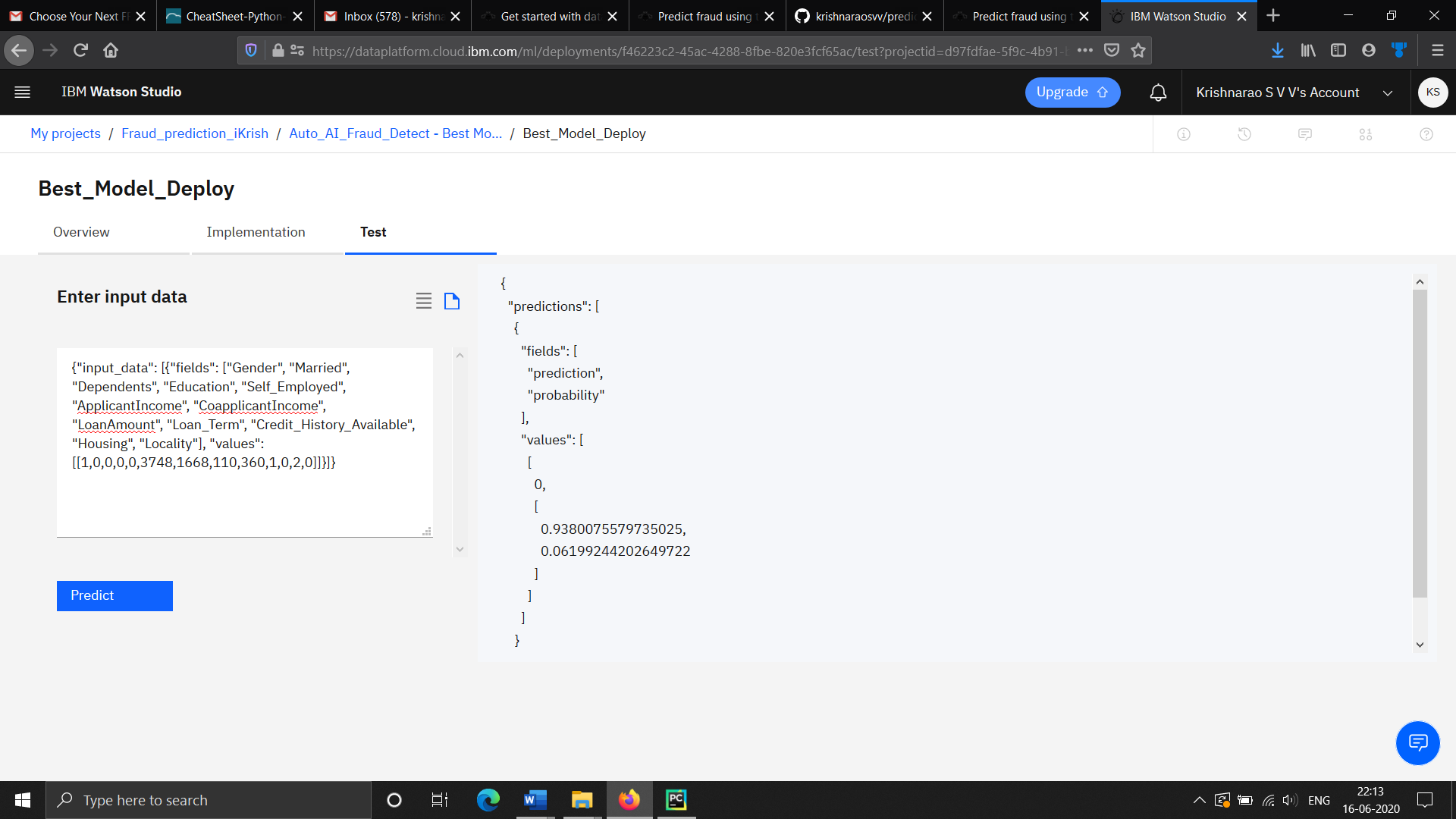
1. Log in to Watson Studio, create a project, and initiate an instance of AutoAI and Cloud Object Storage.
2. Upload the .csv data file to Object Storage.
3. Initiate the model building process using AutoAI and create pipelines.
4. Evaluate different pipelines from AutoAI and select the best model for deployment.
5. Generate accurate predictions by making REST calls to the deployed model.

**Instructions**

Find the detailed steps for this pattern in the [README](https://github.com/IBM/predict-fraud-using-auto-ai/blob/master/README.md). Those steps show you how to:

1. Create an account with IBM Cloud.
2. Create a new Watson Studio project.
3. Add data to the project.
4. Add an asset as AutoAI.
5. Create and define an experiment.
6. Import the .csv file.
7. Run the experiment.
8. Analyze the results.
9. Deploy to IBM Cloud.
10. Test the model.





{

"predictions": [

{

"fields": [

"prediction",

"probability"

],

"values": [

[

0,

[

0.9380075579735025,

0.06199244202649722

]

]

]

}

]

}

{"input\_data": [{"fields": ["Gender", "Married", "Dependents", "Education", "Self\_Employed", "ApplicantIncome", "CoapplicantIncome", "LoanAmount", "Loan\_Term", "Credit\_History\_Available", "Housing", "Locality"], "values": [[1,0,0,0,0,3748,1668,110,360,1,0,2,0]]}]}

