How to use Intel® DAAL Decision Forest via SageMaker web interface

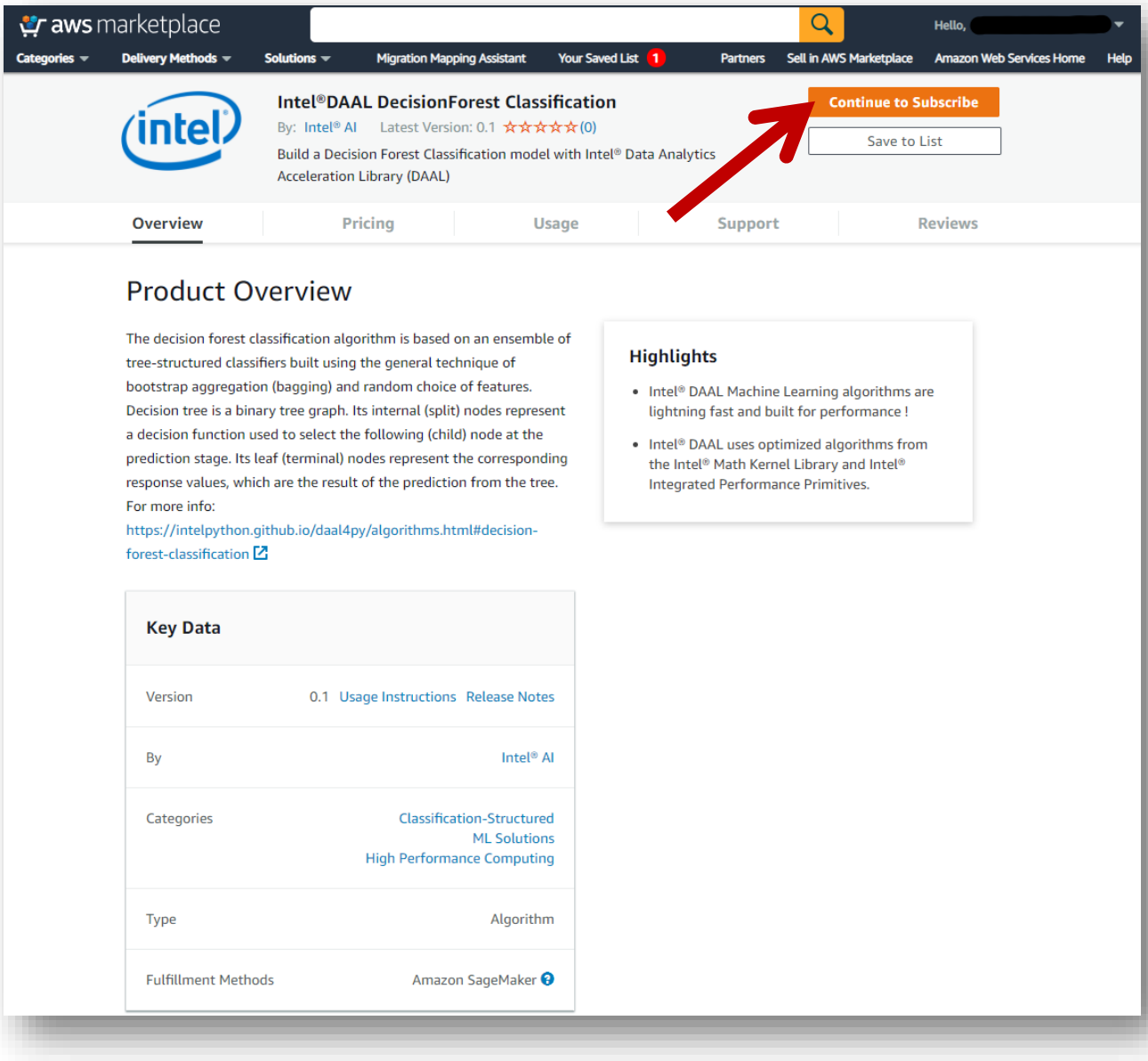
Description of algorithm:

Decision forest classification and regression algorithms based on an ensemble of tree-structured classifiers (decision trees) built using the general technique of bootstrap aggregation (bagging) and random choice of features. Decision tree is a binary tree graph. Its internal (split) nodes represent a decision function used to select the following (child) node at the prediction stage. Its leaf (terminal) nodes represent the corresponding response values, which are the result of the prediction from the tree.

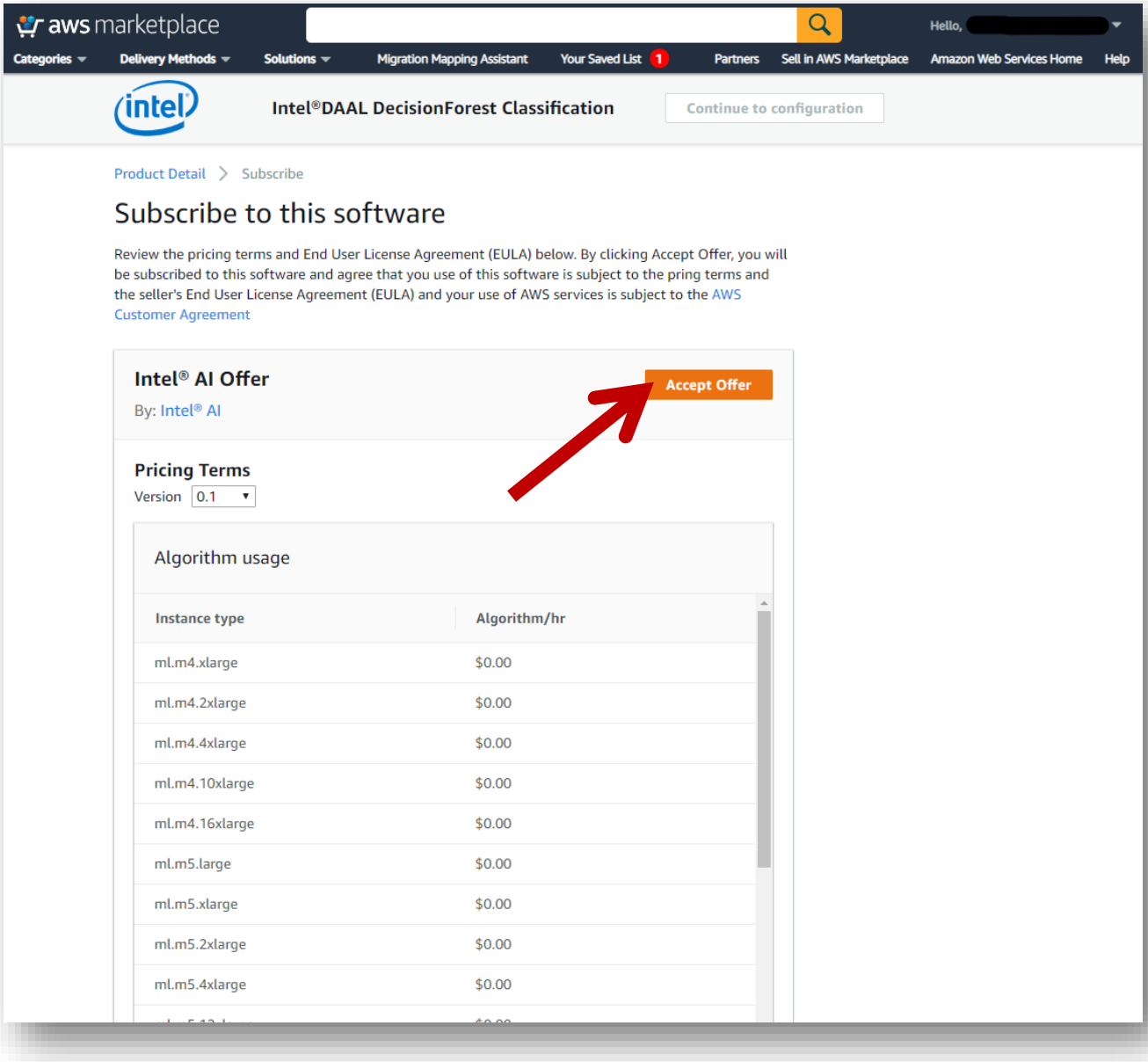
Intel® DAAL developer guide Intel® DAAL documentation for Decision Forest

Instruction:

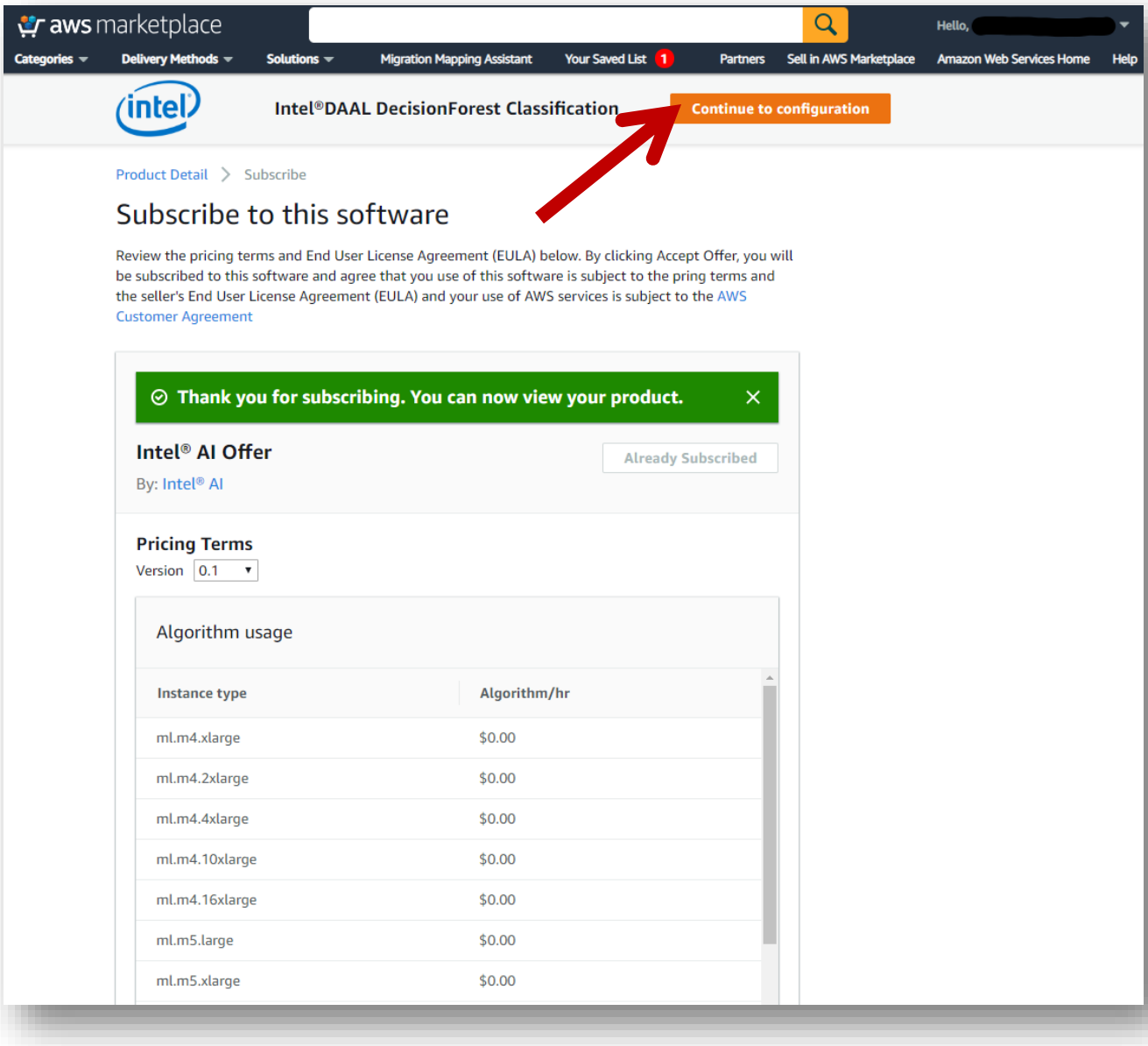
1. Visit page on SageMaker Marketplace and click “Continue to Subscribe”



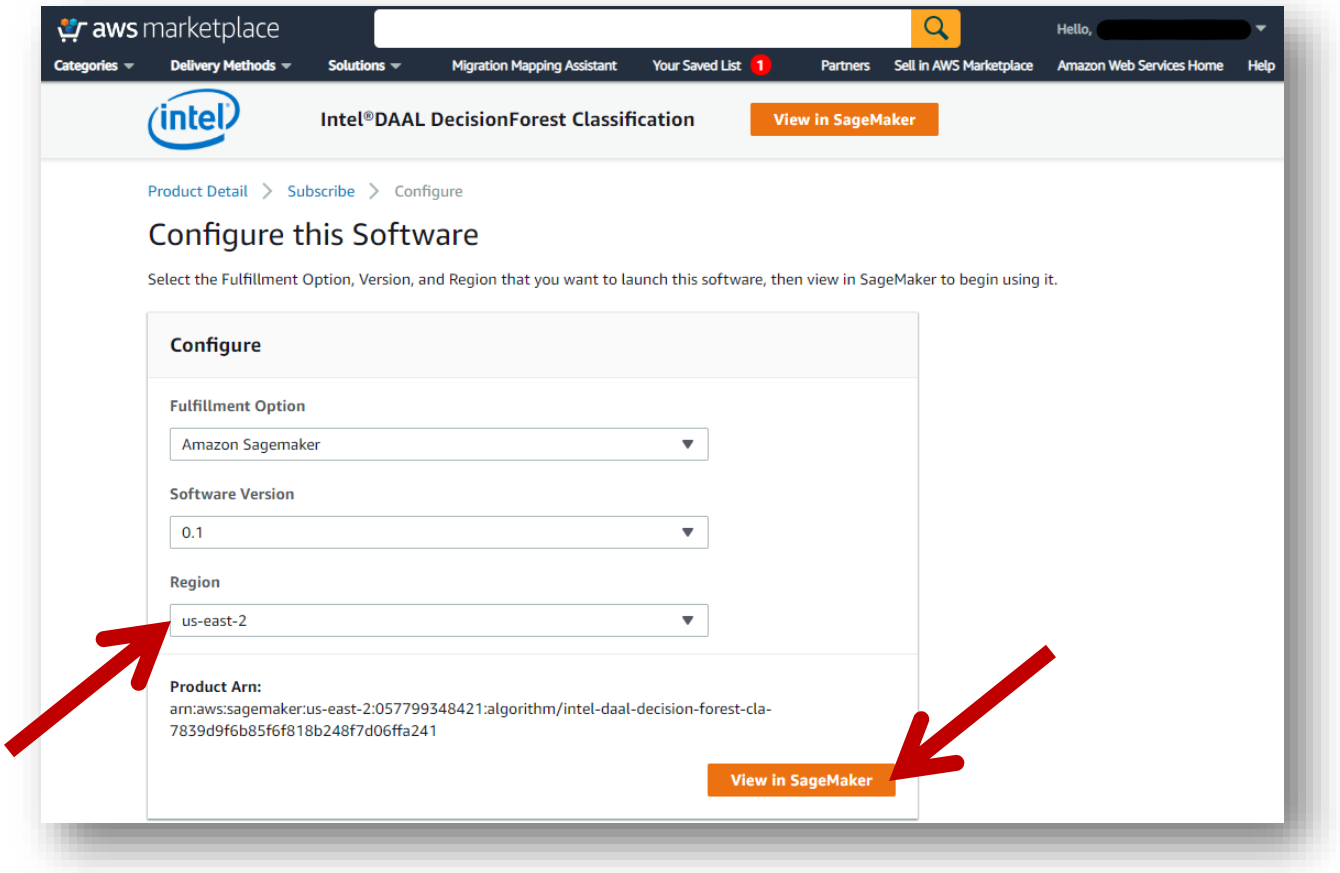
2. Click “Accept Offer” if you agree with EULA at end of page. If you already subscribed on algorithm on Marketplace this step will be skipped.



3. Click “Continue to configuration”



4. Choose the Region and click “View in SageMaker”



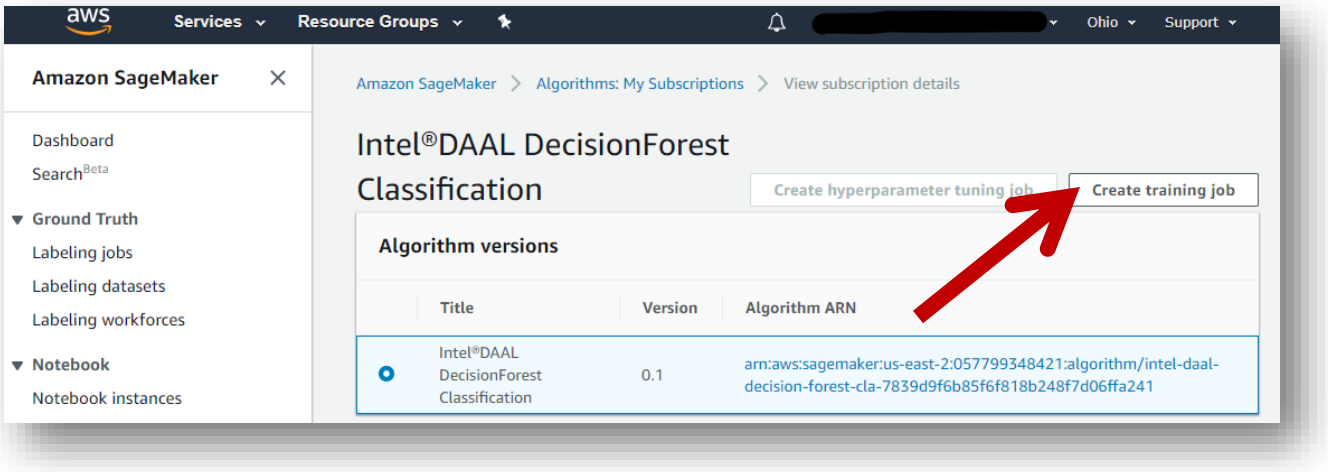
Decision forest algorithm on AWS SageMaker is divided into two stages: training job and getting inference from endpoint.

Training job is computing tree and other values from provided training data.

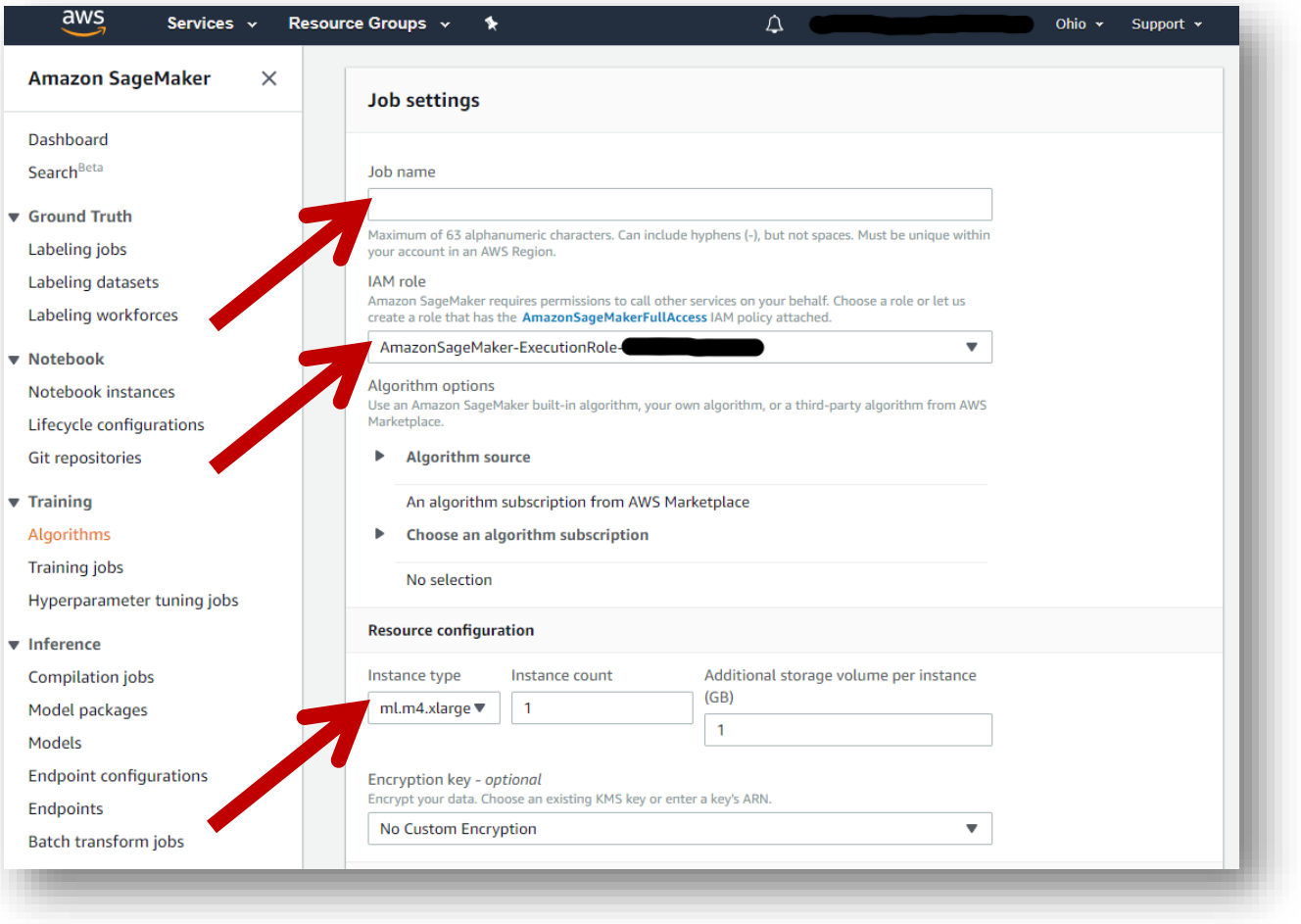
After that, you should create model with computed values and endpoint based on it.

Sending data to endpoint gives you predictions in response.

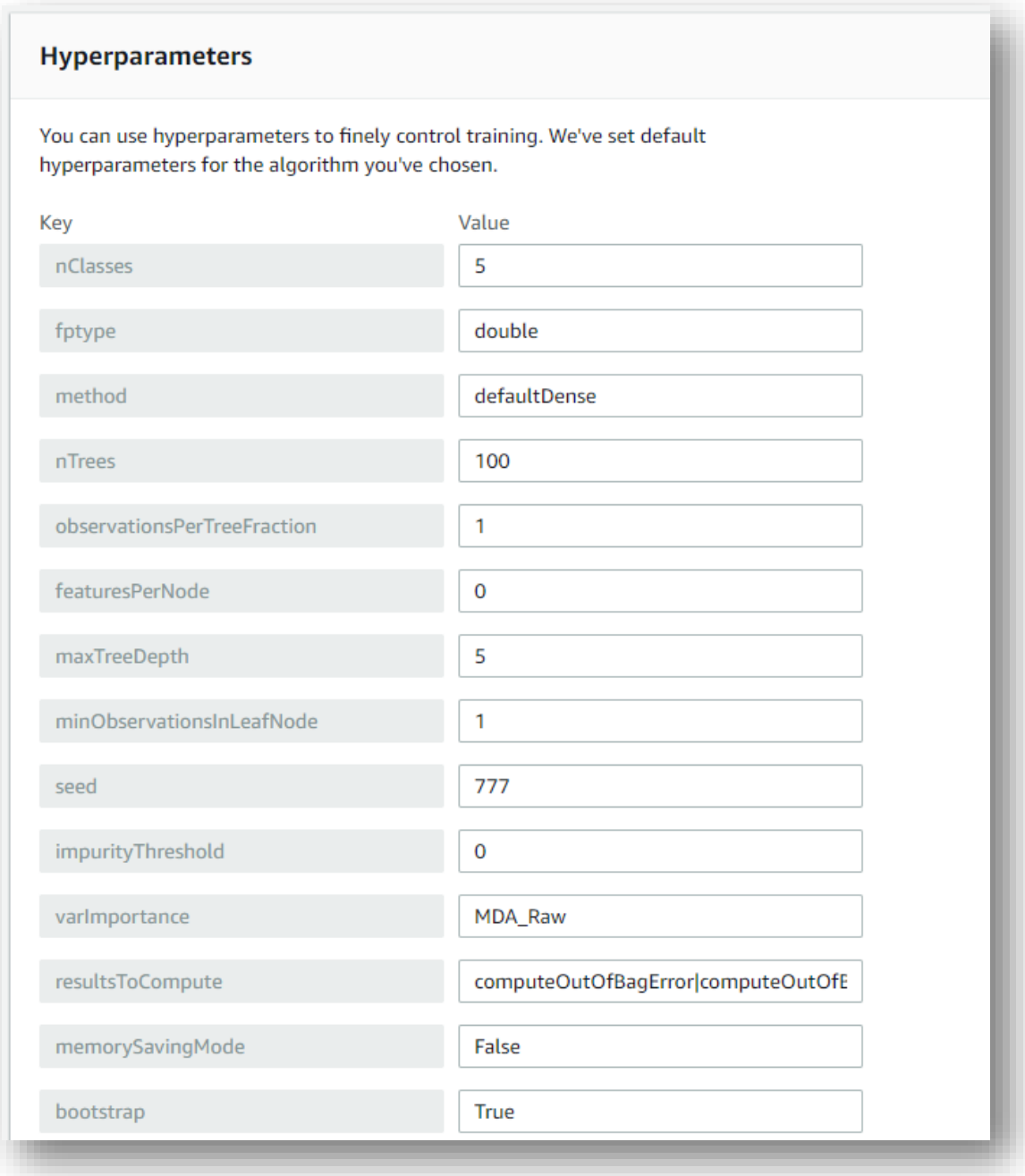
5. Select needed algorithm version and click “Create training job”



6. Type job name, select IAM role and instance type

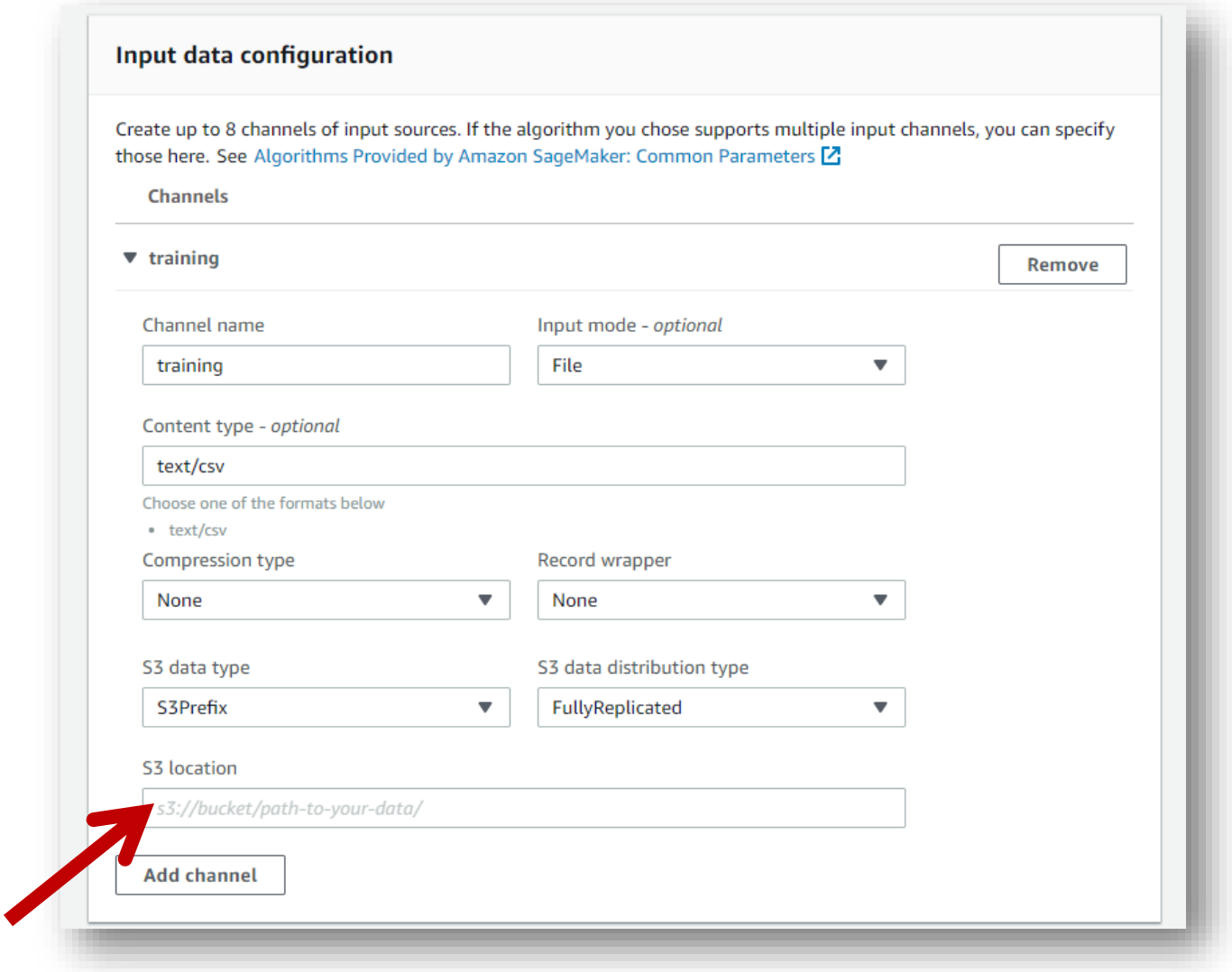


7. Choose hyperparameters

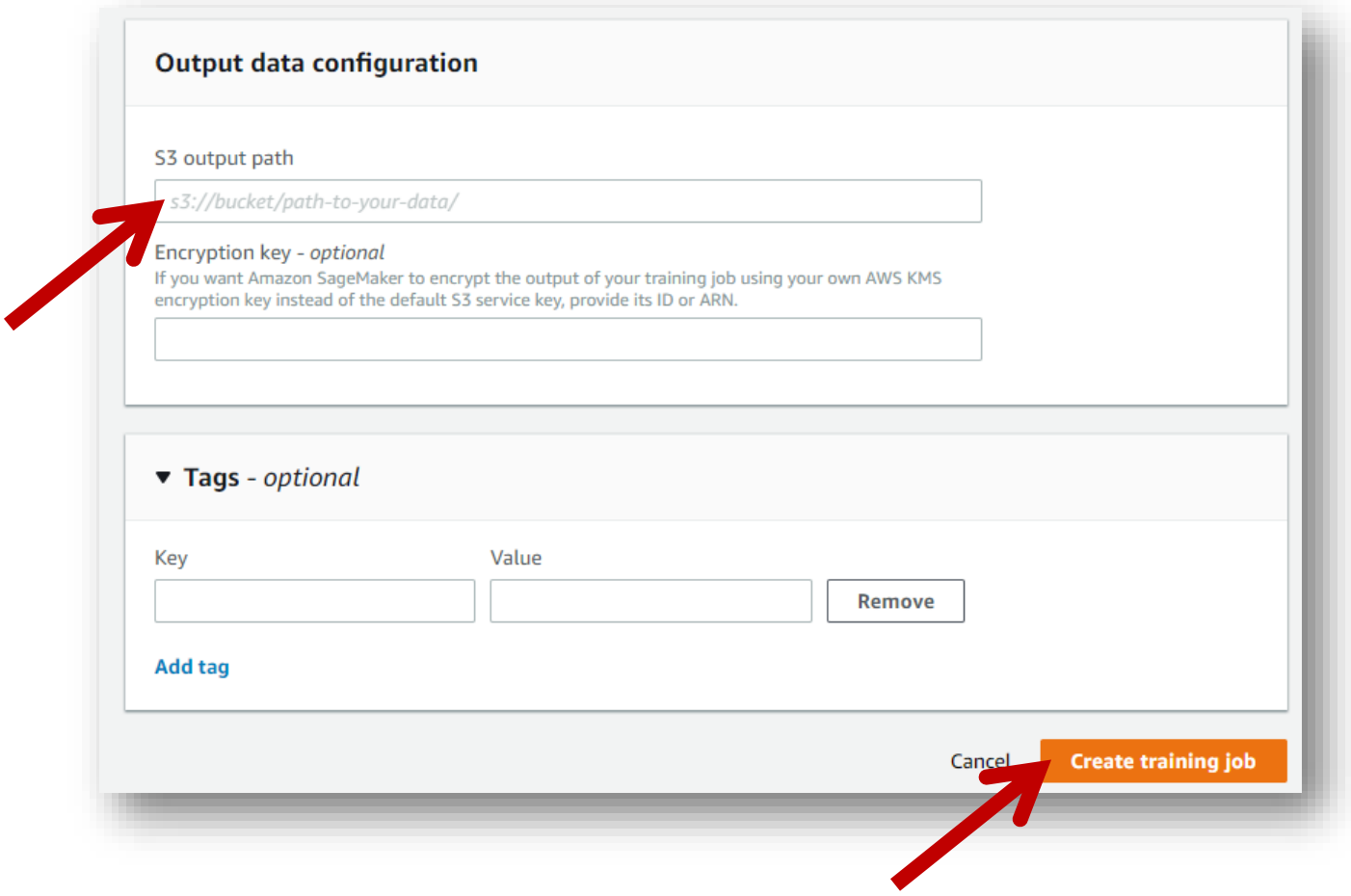


Parameter name	Type	Default value	Description
nClasses	int	2	Number of classes in data (only for classification)
fptype	str	"double"	The floating-point type that the algorithm uses for intermediate computations. Can be "float" or "double"
method	str	"defaultDense"	The only training method supported so far is the default dense method
nTrees	int	100	The number of trees in the forest
observationsPerTreeFraction	int	1	Fraction of the training set S used to form the bootstrap set for a single tree training, observationsPerTreeFraction in (0, 1]. The observations are sampled randomly with replacement
featuresPerNode	int	0	The number of features tried as possible splits per node. If the parameter is set to 0, the library uses the square root of the number of features for classification and (the number of features)/3 for regression
maxTreeDepth	int	0	Maximal tree depth. Default is 0 (unlimited).
minObservationsInLeafNode	int	1 for classification, 5 for regression	The number of neighbors
seed	int	777	The seed for random number generator, which is used to choose the bootstrap set, split features in every split node in a tree, and generate permutation required in computations of MDA variable importance
impurityThreshold	float	0	The threshold value used as stopping criteria: if the impurity value in the node is smaller than the threshold, the node is not split anymore
varImportance	str	“none”	The variable importance computation mode. Possible values: none – variable importance is not calculated MDI - Mean Decrease of Impurity, also known as the Gini importance or Mean Decrease Gini MDA_Raw - Mean Decrease of Accuracy (permutation importance) MDA_Scaled - the MDA_Raw value scaled by its standard deviation
resultsToCompute	str	"none"	Provide one of the following values to request a single characteristic or use bitwise OR to request a combination of the characteristics: computeOutOfBagError, computeOutOfBagErrorPerObservation
memorySavingMode	bool	False	If True, memory saving mode is enabled
bootstrap	bool	False for classification, True for regression	If True, bootstrap is enabled

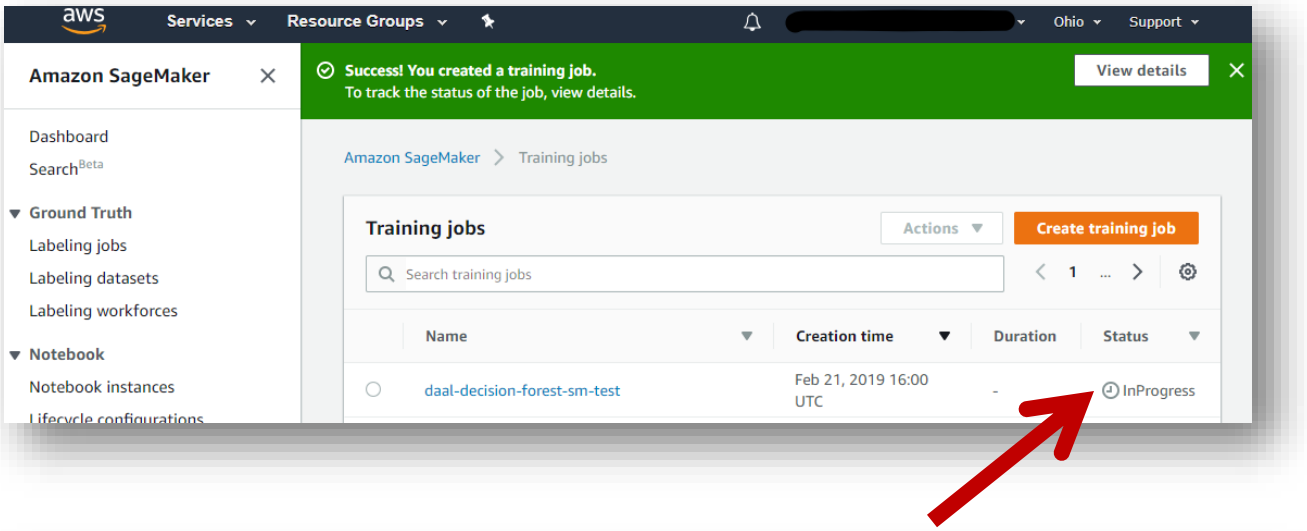
8. Specify S3 location of input data for training

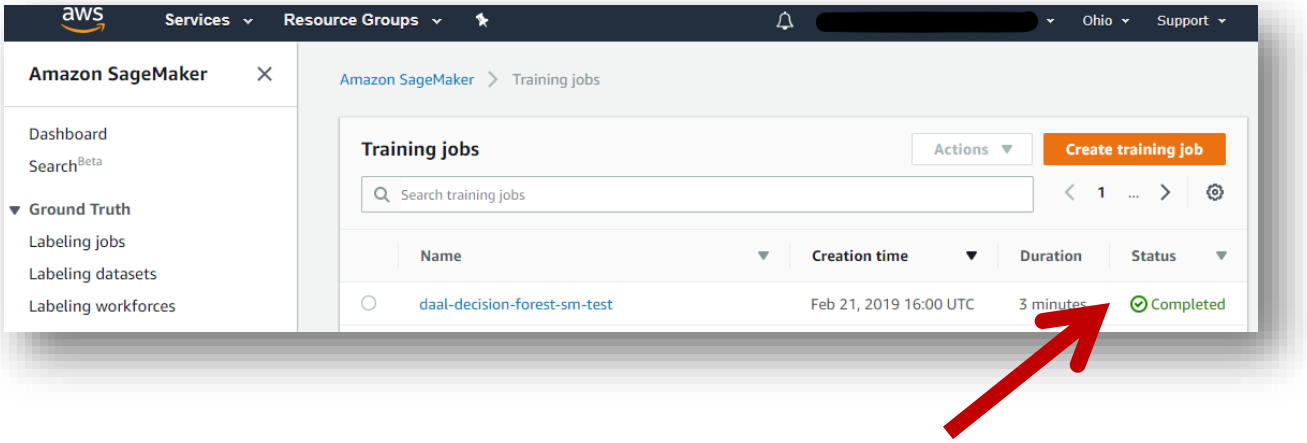


9. Specify S3 output path (model will be stored here) and click “Create training job”

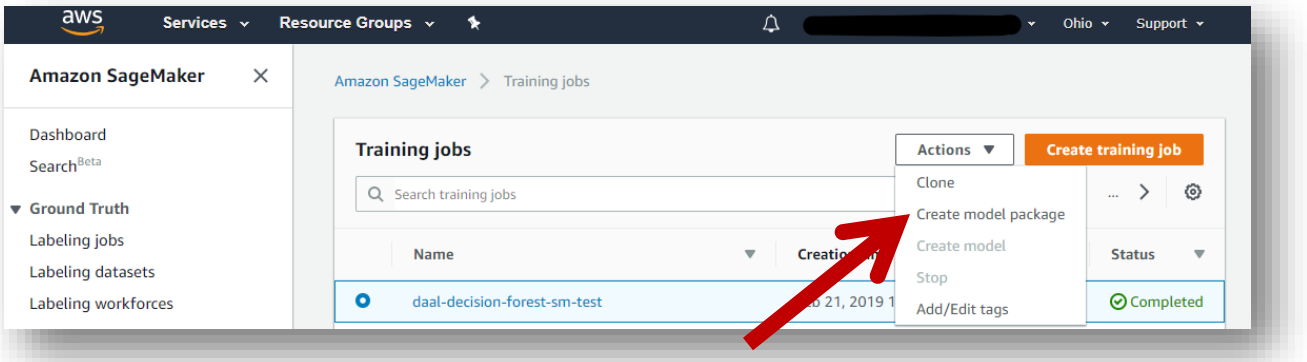


10. Wait until finish of training job

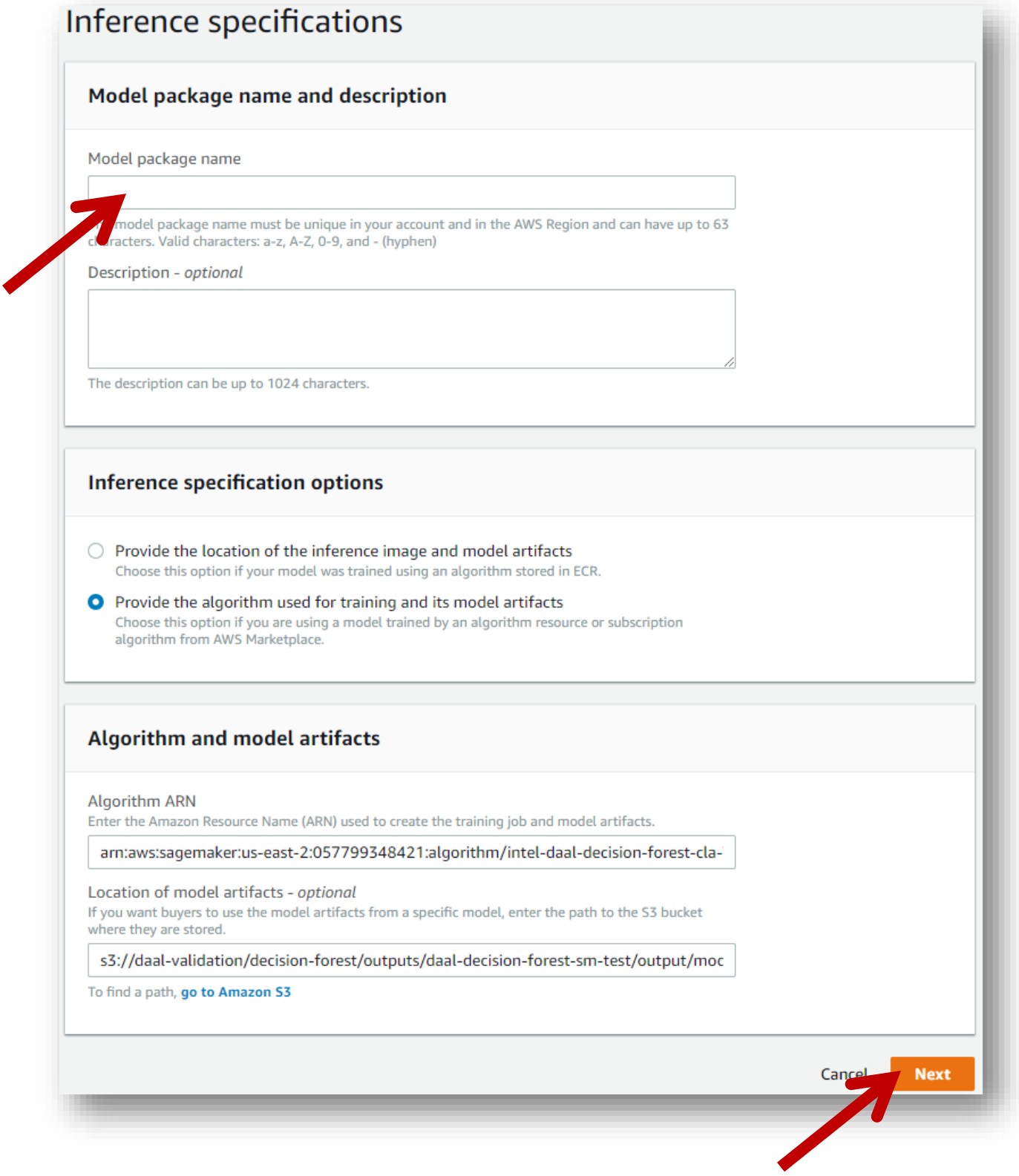




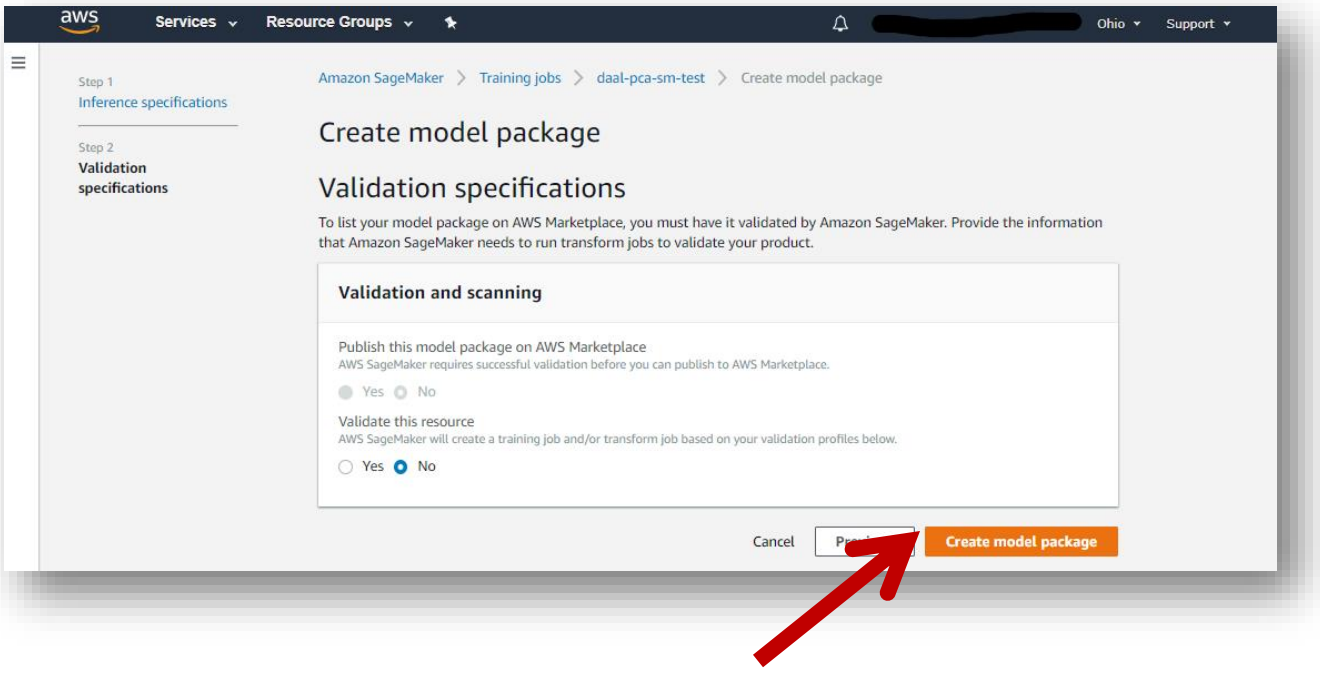
11. Select training job and take action “Create model package”



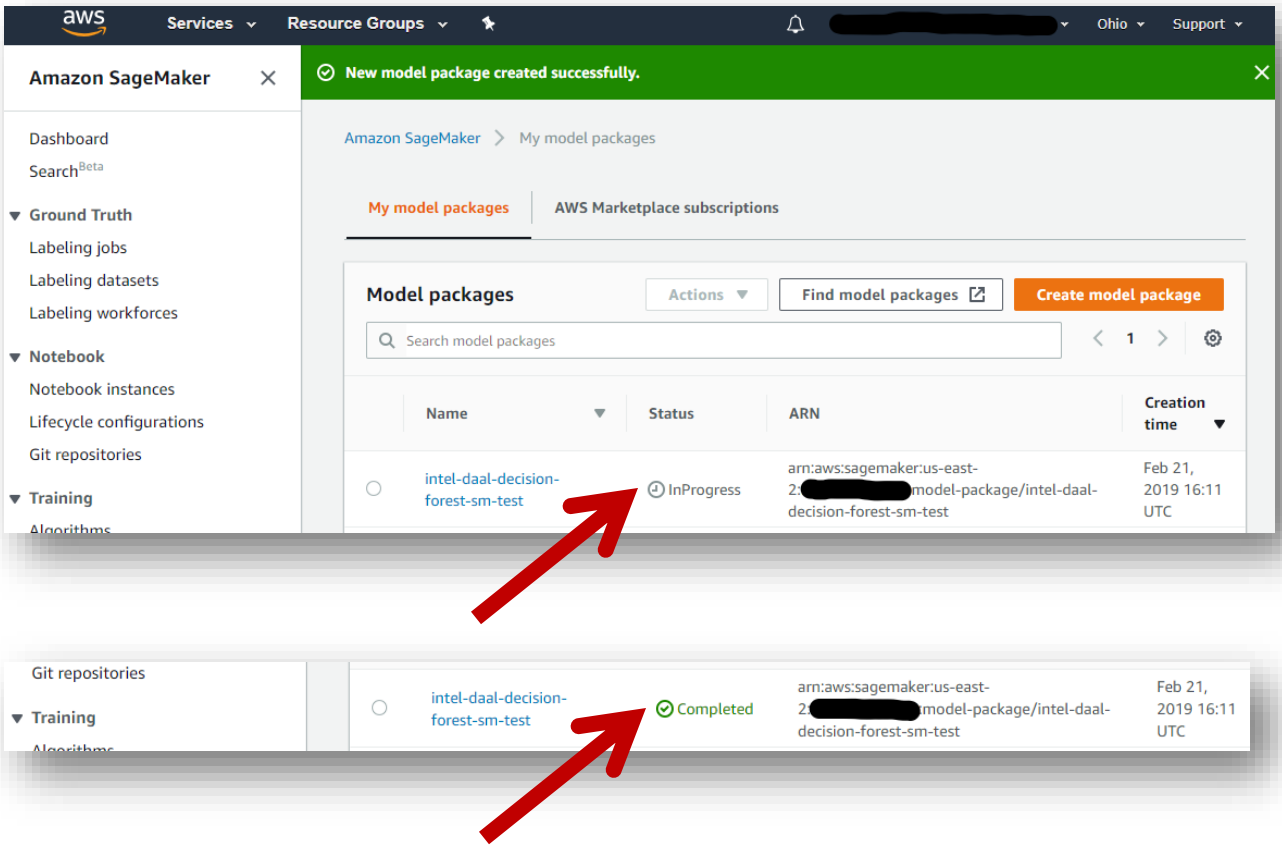
12. Type model package name and click “Next”



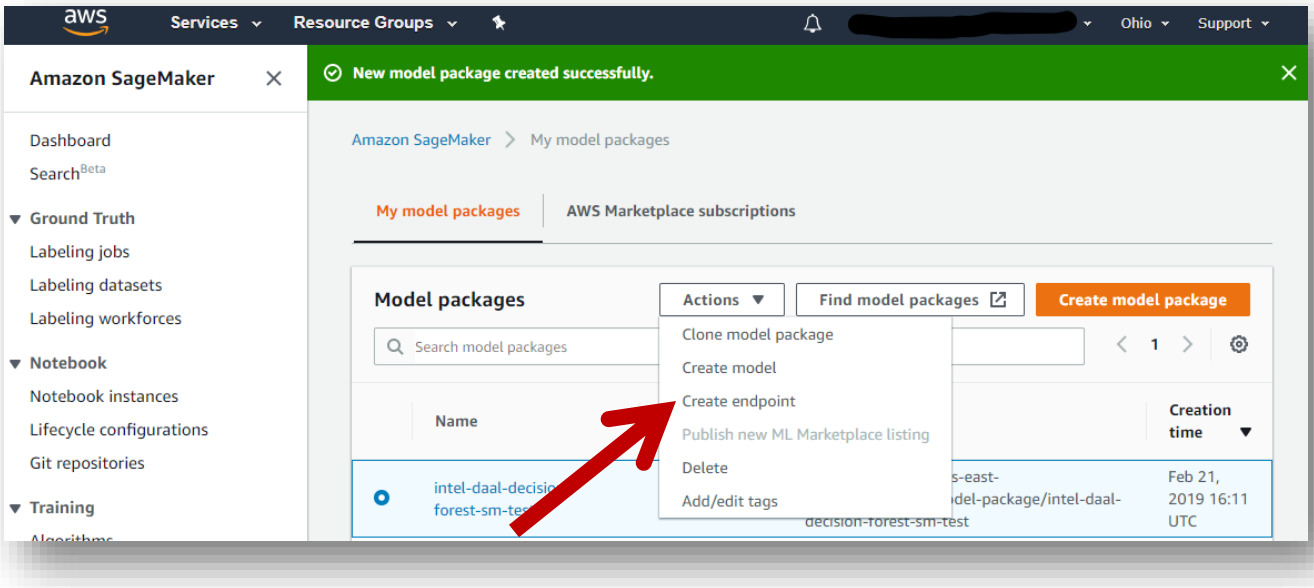
13. Click “Create model package”



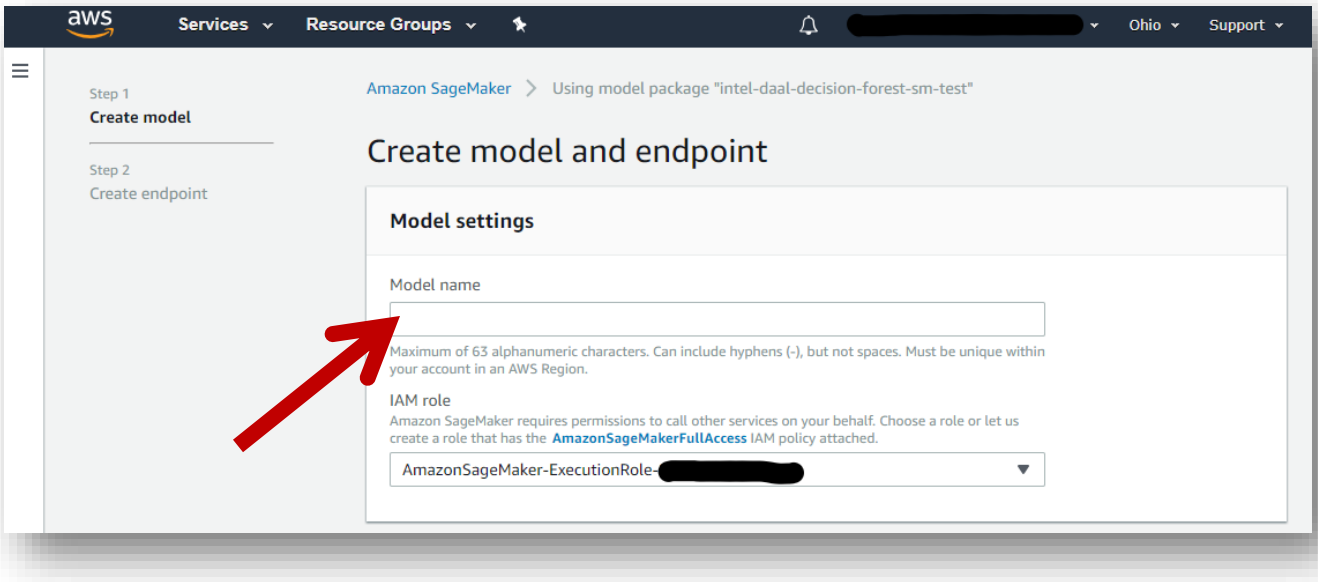
14. Wait until package is created



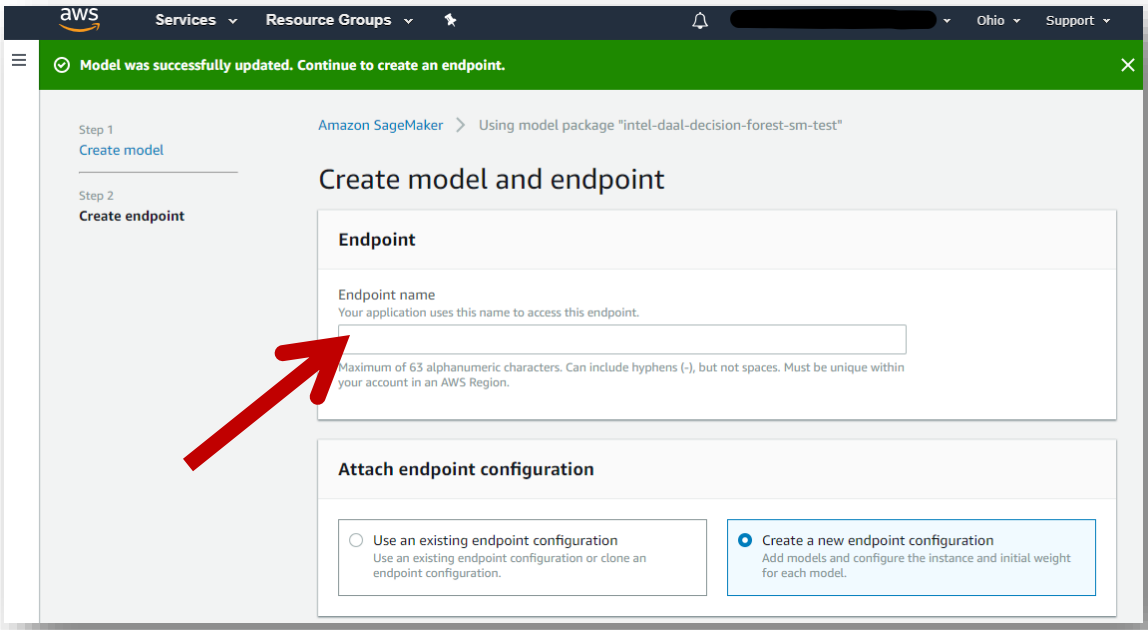
15. Select package and take action “Create endpoint”

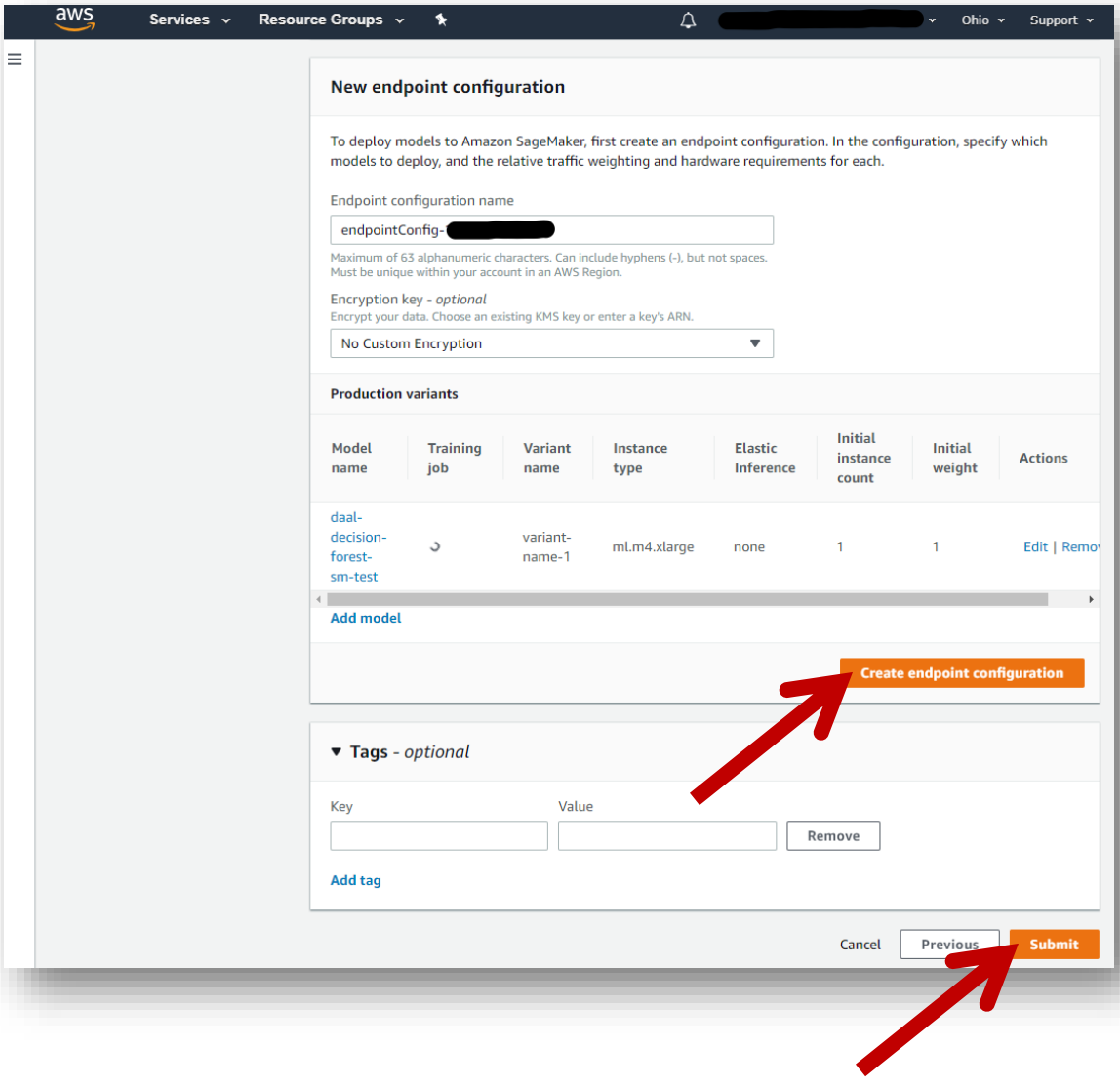


16. Type model name and click “Next”

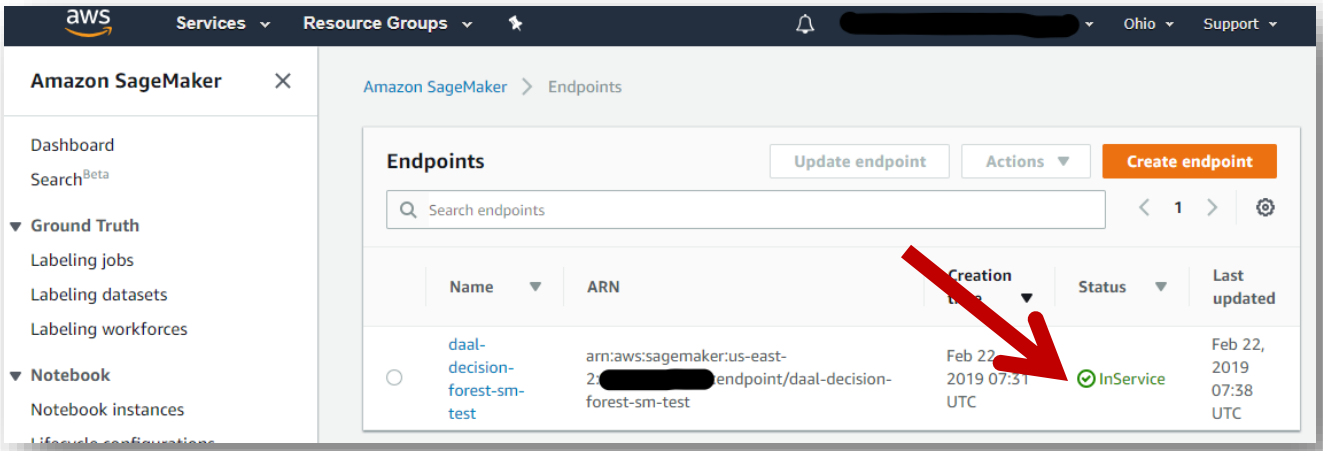


17. Type endpoint name, edit and create endpoint configuration and click “Submit”





18. Wait until endpoint is ready



19. Use AWS CLI to get real-time prediction.

Type command:
aws sagemaker-runtime invoke-endpoint --endpoint-name <endpoint-name> --body "$(cat <prediction_data_file_name>)" --content-type text/csv --accept text/csv <output_data_file_name>

```
(base) ubuntu@ip-172-31-11-84:~$ aws sagemaker-runtime invoke-endpoint --endpoint-name daal-pca-sm-test
--body "$(cat probe_data.csv)" --content-type text/csv --accept text/csv output.txt
```

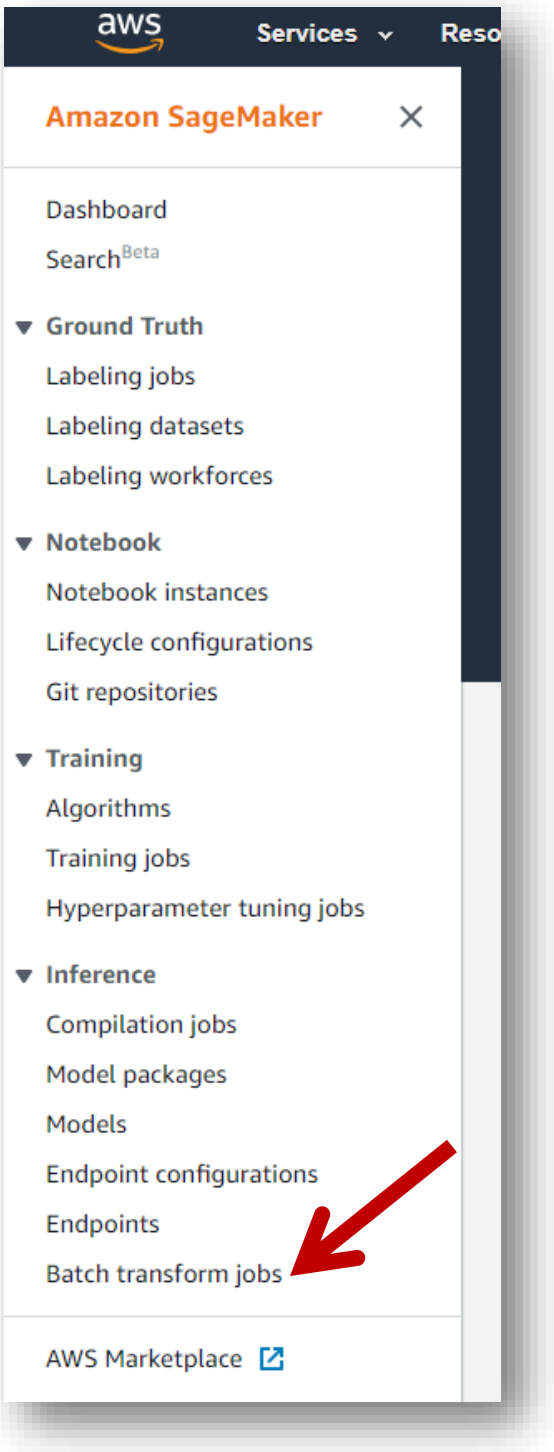
Then, see content of output file:

```
(base) ubuntu@ip-172-31-22-46:~$ cat output.txt
1
4
0
2
3
4
1
1
0
2
4
```

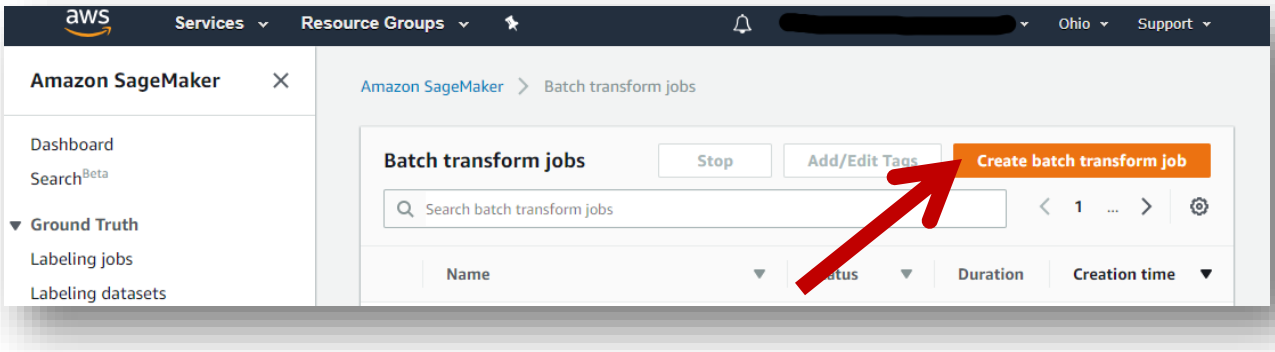
Batch transform job as alternative to endpoint

You can use batch transform job if you need compute predictions once.

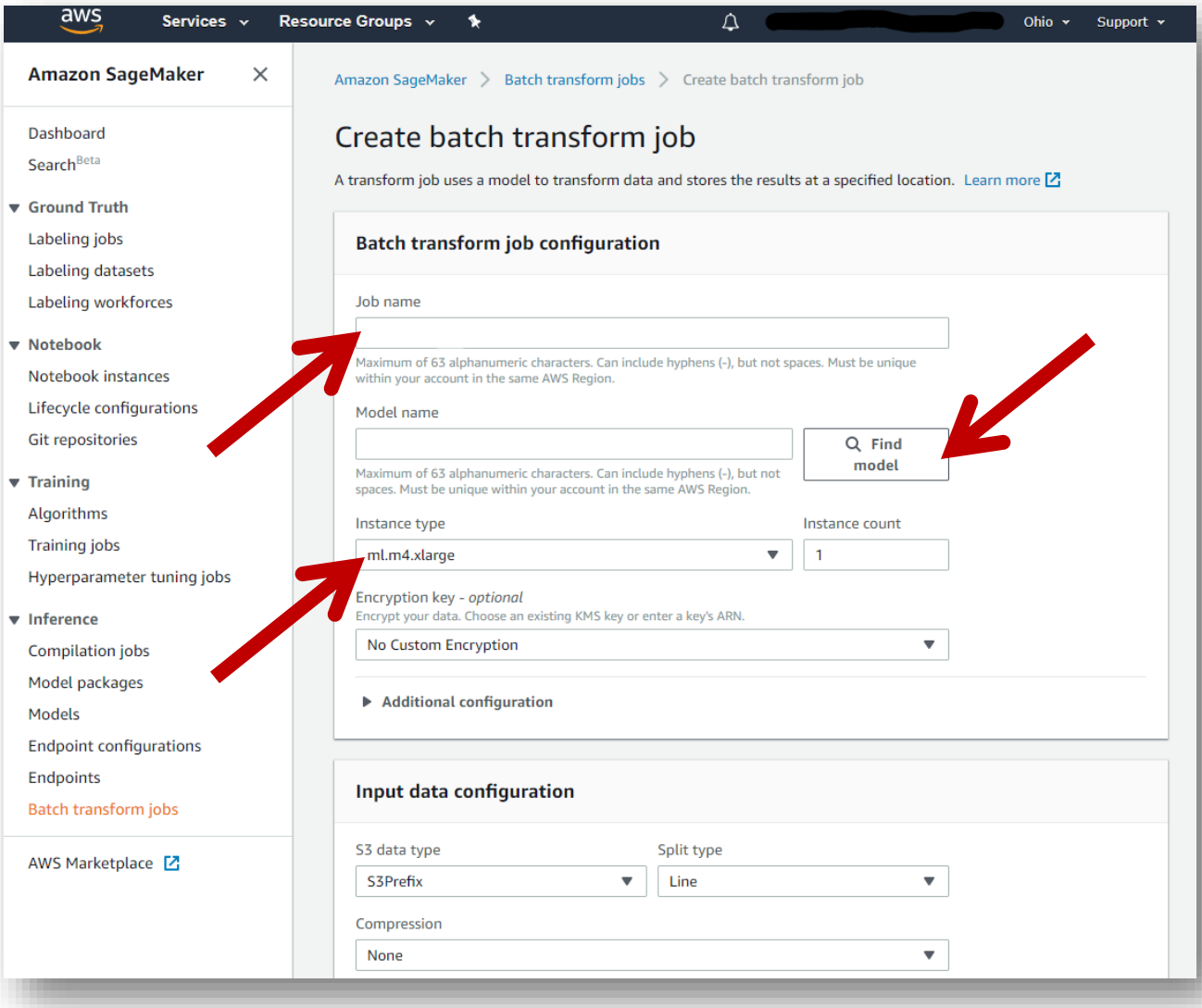
1. Go to “Batch transform job” page



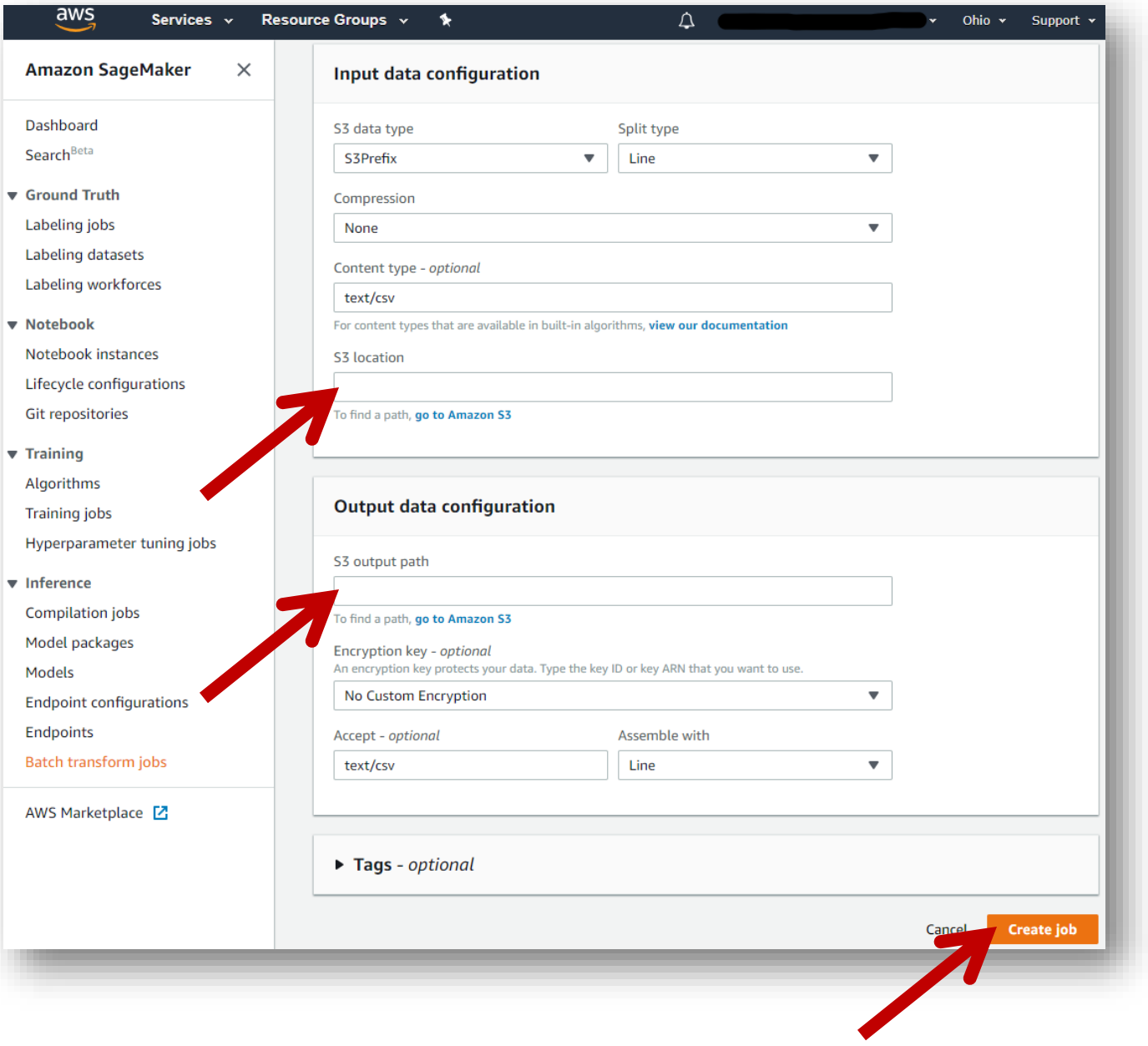
2. Click “Create batch transform job”



3. Enter job name, select previously created model, instance and set instance count to 1



4. Specify S3 location of input data for prediction, S3 output path (predictions will be stored here) and click “Create job”



5. Wait until job is completed and find predictions in previously specified S3 output path