AI Platform online prediction is a service optimized to run your data through hosted models with as little latency as possible. You send small batches of data to the service and it returns your predictions in the response.

about online versus batch prediction (/ml-engine/docs/tensorflow/online-vs-batch-prediction) or read an overview of predi
epts (/ml-engine/docs/prediction-overview).

In order to request predictions, you must first:

- Export your trained model (/ml-engine/docs/exporting-for-prediction) as one or more model artifacts that can be deployed to AI Platform Prediction.
- Deploy your trained model to AI Platform Prediction (/ml-engine/docs/deploying-models) by creating a model resource and version.

AI Platform online prediction is currently available in the following regions:

- `us-central1`
- `us-east1`
- `us-east4`
- `asia-northeast1`
- `europe-west1`

Compute Engine (N1) machine types for online prediction (beta) (/ml-engine/docs/machine-types-online-prediction) are only available in `us-central1`.

To fully understand the available regions for AI Platform training and prediction services, read the guide to regions (/ml-engine/docs/regions).

You make the following important decisions about how to run online prediction when creating the model and version resources:

Resource created	Decision specified at resource creation
Model	Region in which to run predictions
Model	Enable online prediction logging
Version	Runtime version to use
Version	Python version to use
Version	Machine type to use for online prediction

You can't update the settings listed above after the initial creation of the model or version. If you need to change these settings, create a new model or version resource with the new settings and redeploy your model.

When you create a version, you can choose what type of virtual machine AI Platform Prediction uses for online prediction nodes. Learn more about machine types. (/ml-engine/docs/machine-types-online-prediction)

The AI Platform prediction service does not provide logged information about requests by default, because the logs incur cost. Online prediction at a high rate of queries per second (QPS) can produce a substantial number of logs, which are subject to Stackdriver pricing (/stackdriver/pricing) or BigQuery pricing (/bigquery/pricing).

If you want to enable online prediction logging, you must configure it when you create a model resource (/ml-engine/docs/deploying-models#create_a_model_resource) or when you create a model version resource (/ml-engine/docs/deploying-models#create_a_model_version), depending on which type of logging you want to enable. There are three types of logging, which you can enable independently:

- **Access logging**, which logs information like timestamp and latency for each request to Stackdriver Logging.

  You can enable access logging when you create a model resource.

- **Stream logging**, which logs the `stderr` and `stdout` streams from your prediction nodes to Stackdriver Logging, and can be useful for debugging. This type of logging is in beta, and it is not supported by Compute Engine (N1) machine types (/ml-engine/docs/machine-types-online-prediction).

  You can enable stream logging when you create a model resource.

- **Request-response logging**, which logs a sample of online prediction requests and responses to a BigQuery table. This type of logging is in beta.

  You can enable request-response logging by creating a model version resource, then updating that version (/ml-engine/reference/rest/v1/projects.models.versions/patch).

★ **Note:** Do not enable request-response logging if you plan to set up continuous evaluation (/ml-engine/docs/continuous-evaluation/) for your model version. Continuous evaluation configures request-response logging automatically.

You can use the What-If Tool (https://pair-code.github.io/what-if-tool/)(WIT) within notebook environments to inspect AI Platform models through an interactive dashboard. The What-If Tool integrates with TensorBoard, Jupyter notebooks, Colab notebooks, and JupyterHub. It is also pre-installed on AI Platform Notebooks TensorFlow instances.

Learn how to use the What-If Tool with AI Platform (/ml-engine/docs/using-what-if-tool).

The basic format for online prediction is a list of data instances. These can be either plain lists of values or members of a JSON object, depending on how you configured your inputs in your training application. TensorFlow models and custom prediction routines (/ml-engine/docs/custom-prediction-routines) can accept more complex inputs, while most scikit-learn and XGBoost models expect a list of numbers as input.

This example shows an input tensor and an instance key to a TensorFlow model:

The makeup of the JSON string can be complex as long as it follows these rules:

- The top level of instance data must be a JSON object: a dictionary of key/value pairs.
- Individual values in an instance object can be strings, numbers, or lists. You cannot embed JSON objects.

- Lists must contain only items of the same type (including other lists). You may not mix string and numerical values.

You pass input instances for online prediction as the message body for the projects.predict (/ml-engine/reference/rest/v1/projects/predict) call. Learn more about the request body's formatting requirements (/ml-engine/docs/v1/predict-request).

This following section only applies to prediction with TensorFlow.

Binary data cannot be formatted as the UTF-8 encoded strings that JSON supports. If you have binary data in your inputs, you must use base64 encoding to represent it. The following special formatting is required:

- Your encoded string must be formatted as a JSON object with a single key named `b64`. The following Python 2.7 example encodes a buffer of raw JPEG data using the base64 library to make an instance:

  In Python 3, base64 encoding outputs a byte sequence. You must convert this to a string to make it JSON serializable:

- In your TensorFlow model code, you must name the aliases for your binary input and output tensors so that they end with '_bytes'.

Request an online prediction by sending your input data instances as a JSON string in a predict (/ml-engine/reference/rest/v1/projects/predict) request. For formatting of the request and response body, see the details of the prediction request (/ml-engine/docs/v1/predict-request).

If you don't specify a model version, your prediction request uses the default version of the model (/ml-engine/reference/rest/v1/projects.models.versions/setDefault).

Common errors in online prediction include the following:

- Out of memory errors
- Input data is formatted incorrectly
- A single online prediction request must contain no more than 1.5 MB of data. Requests created using the `gcloud` tool can handle no more than 100 instances per file. To get predictions for more instances at the same time, use batch prediction.

Try reducing your model size (/ml-engine/docs/exporting-for-prediction#check_and_adjust_model_size) before deploying it to AI Platform for prediction.

See more details on troubleshooting online prediction (/ml-engine/docs/troubleshooting#troubleshooting_prediction).

- Use batch prediction (/ml-engine/docs/tensorflow/batch-predict) to get inferences asynchronously.
- Get more details about the prediction process (/ml-engine/docs/prediction-overview).
- Troubleshoot problems (/ml-engine/docs/troubleshooting#troubleshooting_prediction) that arise when you request online predictions.