**MindsDB LLMs Models Integrations**

**Hugging Face Models**

In MindsDB it’s possible to use LLMs models without necessarily depending on the use of external APIs like OpenAI. For example, it is possible to integrate Hugging Face models or our own trained models specifically for our needs.

The general query for Hugging Face models will be structured as follows:

| CREATE MODEL huggingface\_model PREDICT target\_column USING  engine = 'huggingface\_engine', -- engine name as created via CREATE ML\_ENGINE  model\_name = 'hf\_hub\_model\_name', -- choose one of PyTorch models from the Hugging Face Hub  task = 'task\_name', -- choose one of 'text-classification', 'text-generation', 'zero-shot-classification', 'translation', 'summarization', 'text2text-generation', 'fill-mask'  input\_column = 'column\_name', -- column that stores input/question to the model  labels = ['label 1', 'label 2']; -- labels used to classify data (used for classification tasks) |
| --- |

Let’s suppose that we want to get information about our dataset **Patient\_Care\_Path\_with\_Doctor\_Notes**, where we have information about the patient and an additional column with notes written by the doctor.

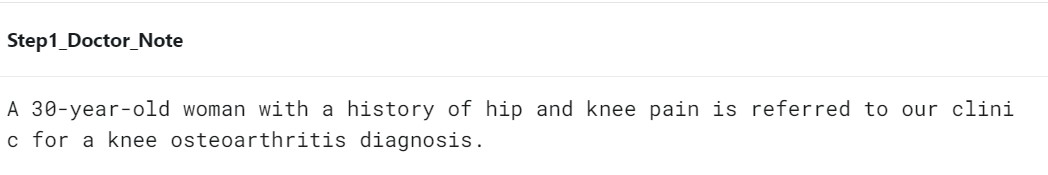
First of all, we should be aware of the data quality and do exploratory data analysis and cleaning. So, when we have the data cleaned and prepared to be analysed we can create an LLM model to answer some specific questions based on the doctor’s notes. Based on the information of this dataset, we can try to get a summary of the doctor’s information for a specific patient.

For getting a summary, we can use the model of Google Pegasus, that removes/masks important sentences from an input document and generates them together as one output sequence from the remaining sentences, similar to an extractive summary.

| CREATE MODEL mindsdb.hf\_peg\_sum\_20 PREDICT Step1\_Doctor\_Note USING  engine = 'huggingface',  task = 'summarization',  model\_name = 'google/pegasus-xsum',  input\_column = 'Step1\_Doctor\_Note',  min\_output\_length = 5,  max\_output\_length = 20; |
| --- |

To see the results, the following query should be executed:

| SELECT h.\*, t.Step1\_Doctor\_Note AS input\_text FROM valencia\_medical\_data.patient\_care\_path\_with\_selective\_doctor\_s\_notes AS t JOIN mindsdb.hf\_peg\_sum\_20 AS h LIMIT 2; |
| --- |



The final step will be to visualise the data. The prediction results could be integrated in dashboards or used in front-end applications to show predictions to the final user.

To sum up, the model could summarise medical notes or provide specific insights from the text notes, predict future diagnosis if you have enough data and automate responses which can be very useful in healthcare.

More detailed examples about how does Hugging Face models work are shown here: [Usage Examples of Hugging Face Models - MindsDB](https://docs.mindsdb.com/use-cases/data_enrichment/hugging-face-examples)

**OpenAI Models**

In addition and as said before, it’s possible to connect with OpenAI (or other cheaper AI integrations) by creating a model based on LLM. The code is the following:

| CREATE MODEL model\_name PREDICT predict\_column USING  engine = 'openai',  model\_name = 'openai\_model\_name',  prompt\_template = 'Prompt with {{variables}}'; |
| --- |

To use OpenAI LLMs models, it’s necessary to have an API key provided by OpenAI. This key allows the user to authenticate and authorize the requests made to OpenAI services.

Once the model is created as seen above, you can try executing different consults to see how the model responds, by making sure that it meets your requirements.

Let’s suppose we continue using the dataset **Patient\_Care\_Path\_with\_Doctor\_Notes**. This time, the engine would be 'openai'

| CREATE MODEL medical\_notes\_model PREDICT desired\_result USING  engine = 'openai',  model\_name = 'text-davinci-003',  prompt\_template = 'Based on doctor's notes: {{doctor's notes}}, answer: {{question}}'; |
| --- |

desired\_result can be any type of objective that we would want to predict or analyse, for example, their diagnosis, suggested treatment or a forecast.

Some use-case examples could be the following:

* **Make specific questions**

| SELECT desired\_result FROM medical\_notes\_model WHERE question = '¿What is the suggested diagnosis based on the notes?'; |
| --- |

This will allow the model to analyze the notes and provide an answer based on the LLM context

* **Generate notes or predictions for new patients**

| SELECT \* FROM medical\_notes\_model WHERE doctor\_notes = 'Patient with fever symptoms and dry coughing...'; |
| --- |

It’s possible to add new notes or partial information of a patient and let the model generate predictions, recommendations or summaries based on learned patterns from the existing notes.

The model will be able to predict possible diagnosis, suggested treatments or any other desired metric depending on the training.

For more information about the use of the OpenAI Key see: [The LLM() function](https://docs.mindsdb.com/mindsdb_sql/functions/llm_function#the-llm-function)