Building, Training, and Validating Models in Microsoft Azure

by Bismark Adomako

Start Course

Bookmark Add to Channel Download Course

Table of contents Description **Transcript** Exercise files Discussion Related

Course Overview

Course Overview

Hi everyone. My name is Bismark Adomako, and welcome to my course, Building, Training, and Validating Models in Microsoft Azure. I'm a data engineer at Ecobank eProcess International S.A. This course gives Microsoft Azure data scientists a road map on how to build, train, and validate machine learning models in Azure. The course will try to predict if a flight will be delayed more than 15 minutes with the given data. Some of the major topics that will be covered includes creating a hypothesis, identifying features from raw data, building a model, and then monitoring and managing the model performance. By the end of this course, you will know how to use Microsoft Azure Machine Learning services to build, train, and validate models. Before beginning the course, you should be familiar with intermediate level of understanding data science concepts such as supervised classification. I hope you join me on this journey to learn more about Azure Machine Learning services with the Building, Training, and Validating Models in Microsoft Azure course, at Pluralsight.

Creating a Hypothesis

Overview

Hi, my name is Bismark Adomako, and welcome to the first model of our course where we create a hypothesis on which our model performance will be judged. We cannot take a single step forward in any inquiry unless we begin with a suggested explanation or solution of difficulty which are ______. Such ______ explanations are suggested to us by something in the subject matter and by our previous knowledge. When they are from later ______ propositions, they are called hypothesis. The word hypothesis consists of two words, hypo and thesis, Hypo means tentative or subject to the verification, and thesis means statement about the solution of the problem. Therefore, a hypothesis can be defined as a tentative statement about the relationship between two or more variables, and it is a specific, testable prediction about what you expect to happen in your study. In this module, you will create a concrete hypothesis by going through some key concepts, such as identifying factors of a well-defined hypothesis, contrasting hypothesis under test, and a null hypothesis. We will also identify the key performance indicators required by the hypothesis. Then you define the hypothesis acceptance criteria. And finally, we differentiate the hypothesis generation and the hypothesis confirmation. Remember, a hypothesis does not have to be correct. While the hypothesis predicts what the research expects to see, the goal of the research is to determine whether this guess is right or wrong.

Factors of a Well Defined Hypothesis

Obviously, an undefined or ill-defined concept makes it difficult, or, rather impossible, for us to test a hypothesis, as there would not be any standard basis for us to know how _____ there. In this lesson, we identify factors of a well-defined hypothesis. First of all, the hypothesis should be conceptually clear. The concept of using the hypothesis should be clearly defined, not only formally, but also, if possible, operationally. Formal definition of the hypothesis will clarify what a particular concept stands for, while the operational definition will leave no ambiguity about what will constitute the empirical evidence or indicator of the concept on the plane of reality. Also, the hypothesis should be specific. No vague or value judgment attempts should be used in this formulation of a hypothesis. It helps to increase the validity of our result because the more specific the statement or prediction, the smaller the probability that it will actually be born out of a result of mere accidents or chance. You must remember that the narrower the hypothesis, the more testable it is. Again, the hypothesis should be empirically testable. It should have empirical reference so that it will be possible to deduce certain logical deductions and inferences about it. Such statement as criminals are no worse than businessman, capitalists exploit their workers, bad parents beget bad children, bad homes breed criminality, or pigs are named because they are so

______ can hardly be usable hypotheses, as they do not have any empirical reference for testing their validity. In other words, we should avoid themes loaded with values or beliefs or what's ______ or attitude in our connotations in our hypothesis. Lastly, our hypothesis should be related to available techniques. If you are ignorant of the available techniques, it makes us weak in formulating a workable hypothesis. A hypothesis therefore needs to be formulated only after due thought has been given to the methods and the techniques that can be used for measuring the concept or variables incorporated in the hypothesis. In our next lesson, we will talk about the alternative and the null hypotheses. But before that, think about how will they find this hypotheses? Will a flight to be delayed more than 15 minutes?

Alternative Hypothesis and Null Hypothesis

In this lesson, we will contrast the alternative hypothesis from the null hypothesis, and explain why we give special consideration to the null hypothesis. The null hypothesis has said that there is no difference between the sample statistics in the population parameter under consideration, hence the word null, which means invalid, void or amounting to nothing. The null hypothesis proposes no relationship or difference between two variables, however, the alternative hypothesis proposes a relationship between two or more variables. Also, the null hypothesis is a conventional approach to making a prediction. It involves a statement that says that there is no relationship between two groups that a researcher compares on a setting variable. On the other hand, the alternative hypothesis is an unconventional approach to making a prediction. The null hypothesis is denoted by the symbol H0, while the alternative hypothesis is symbolized by H1. For example, if you are interested in examining the relationship between music and emotion, the null hypothesis will state that music at a fast tempo rated the same in happiness by the participants, while the alternative hypothesis will state that music at a fast tempo is rated by participants as being happier than music at a slow tempo. The two hypotheses we propose to test must be mutually exclusive. That is, when one is true, the other must be false. And we see that they must be exhaustive. They must include all possible occurrences. We give special consideration to the null hypothesis. This is due to the fact that the null hypothesis relates to the statement being tested, whereas the alternative hypothesis relates to the statement to be accepted if or when the null is rejected. The final consideration once the test has been carried out is always given in terms of the null hypothesis. We either reject H0 in favor of H1 or do not reject either. We never conclude, reject H1 or even accept H1. If you conclude, do not reject H0. This does not necessarily mean that the null hypothesis is true, it only suggests that there is not sufficient evidence against H0 in favor of H1. Rejecting the null hypothesis then suggests that the alternative hypothesis may be true.

There are two types of mistakes which are possible while testing the hypothesis. In this table considering being sick and being well, when you're sick and then the doctor confirms that you're sick, that is a right prediction, but when you're sick and then the doctor forcefully says that you are okay, that is a wrong prediction, and that's what we call a Type I error. When you are well, and then the doctor predicts that you are sick, that is a wrong prediction, and that is termed as a type II error. However, when you are well and then the doctor predicts that you are well, that is also a right prediction. A Type I error occurs when the null hypothesis is wrongly rejected. A Type I error will occur if the doctor predicts that you are well, while you are actually sick. A Type II error occurs when the null hypothesis is not rejected when it is in fact false. For example, in being sick, when you are well and the doctor predicts that you are sick, that is a Type II error. Bringing it back to our hypothesis, will our flight be delayed more than 15 minutes, what do you think a Type I error will be and what do you think a Type II error will be? I leave this to you while we move to the next lesson where we talk about how to identify key performance indicators required by the hypothesis.

Key Performance Indicators Required

In this lesson, we'll be identifying the key performance indicators required by our hypothesis. But before that, let us to look at the key stages in identifying KPIs for our hypothesis. First of all, we should have a predefined hypothesis, which we already have. We should also have a qualitative or quantitative measurement of the results and comparison with a set hypothesis. This will tell us how well our model is doing. We also need to investigate the variance and tweak some processes. In addition, the KPI we should be defining should be smart. That is, it should be specific to the papers for the research. It should be measurable to really get a value out of the hypothesis. It should also be relevant to the course of a research. Lastly, it should be time bound, which means the value or outcomes are shown for a predefined and relevant timeframe. Hence, the key performance indicators required for our hypothesis should predict if a flight would be delayed more than 15 minutes. It should also state whether the prediction was true or false. Was this prediction relevant to improving flight processes? What is the score of the model to be created? Here, we seek to decrease the type II error. Lastly, we should be able to make predictions within 15 minutes or less before the proposed flight departure. We do not want to create a model which makes a prediction after a flight has been delayed.

Hypothesis Acceptance Criteria

We'll be defining the hypothesis acceptance criteria. To set the criteria for an acceptance, we state the level of significance for our model. This is similar to the criterion that juries use in a criminal trial. The jurors decide whether the evidence presented shows guilt beyond a reasonable doubt. This is the criteria. Likewise, in hypothesis testing, we collect data to show that the null hypothesis is not true based on the likelihood of selecting a sample mean from a population. When the probability of a flight delaying is less than 65%, if the null hypothesis were true, then we conclude that the sample we selected is too unlikely, and so we reject the null hypothesis. We'll also be expecting a higher model performance or accuracy. After training our model, we will be choosing a model outcome with the highest accuracy. This will show us how well the model predicts true values. We'll make a very short, but very important differentiation between the hypothesis generation and the hypothesis confirmation. Generally, in hypothesis generation, inductive reasoning uses the data to generate ideas, whereas in hypothesis confirmation, deductive reasoning begins with the idea and uses the data to confirm or negate the idea. Thus, there is no magic number where inductive reasoning suddenly becomes deductive reasoning. Also, in hypothesis generation, an observation can be used many times, whereas in hypothesis confirmation, an observation can be used only once.

Summary

In this module, we talked extensively about how to identify factors of a well-defined hypothesis. We then went ahead to contrast the hypothesis on a test from the null hypothesis. We also identified key performance indicators required by our hypothesis. We then identified a hypothesis acceptance criteria, and then we differentiated the hypothesis generation and the hypothesis confirmation. In our next module, we'll be performing more data engineering rules by exploring the data further. You'll describe methods of splitting data into training and test it. You'll also be doing an introduction of Azure Machine Learning services and why it's a game changer for data scientists and data engineers.

Sourcing and Transforming Data Relevant to a Hypothesis

Overview

Hello, my name is Bismark Adomako, and I welcome you to this module of our course where we source and transform data relevant to our hypothesis. In this module, we will take a brief overview of what Azure Machine Learning Services is and get it set up. We'll also discuss why it is a game changer for data scientists and data engineers. We will take a brief look at the data we are using from Azure Machine Learning datasets. We will then conduct a brief data exploration using PowerBI. We will also discuss why and how we need to split the data for training and testing the module. Lastly, we will look at how to use Azure ML components to transform data. Come with me as we explore and experience Azure ML Services.

Azure Machine Learning Services Overview

The Azure Machine Learning service provides a cloud-based environment you can use to develop, train, test, deploy, manage and train machine learning models. It fully supports open-source technologies so you can use tens of thousands of open-source Python packages with machine learning components such as TensorFlow and Scikit-learn. Rich tools such as Jupyter Notebooks or the Visual Studio code tools for AI make it easy to interactively explore data, transform it, and then develop and test models. Azure Machine Learning will help simplify building your machine-learning models by using the automated machine-learning capabilities, or if you want to build your own model, you can easily scale out in the cloud, using Python as the case, using any of the open-source frameworks, and you can manage the end-to-end workflow using Azure Machine Learning pipelines, which is like DevOps for machine learning. It also helps you easily deploy your trained models to the cloud and to the Edge. Now the end-to-end workflow includes preparing the data in the cloud using something like Azure Databricks, and then using your own Python code in an ID of its choice, using the Python SDK to train our model and keep track off our metrics, and then we can restart modeling to deploy that to the cloud or to the Edge and orchestrate the whole thing using Azure Machine Learning pipelines. In this demo, you will first take a look at how to set up our Azure Machine Learning workspace, and also copy this up with the Azure Databricks workspace. This is because we'll be writing our code in Azure Databricks. Remember, you can use rich tools such as Jupyter Notebooks or Visual Studio code to do modern development as well. Now let's see how setting it up looks like. Now in our Azure workspace, we'll create a resource. It's an AI or machine-learning resource so select, and then select Machine Learning Resource. We fill in our workspace name, select the subscription, and then the resistance group. And then we set our location to West Europe, because our resource group is located in West Europe. We review and then recreate, but we have an error. Our workspace edition was not selected, and it cannot be empty. So the best thing to do is to go back to the main tab and then select the Workspace edition. We can either select Basic workspace edition or Enterprise one. In this case, we are using the Enterprise one. We then review and create the workspace. As the workspace is created, we go to explore from the resource group where the workspace was created. We select our Pluralsight workspace, and then we launch our Machine Learning Studio. We sign in, and then we fill out our directory and then our subscription again. So in this case you use the default directory, and then we select our subscription, Pluralsight. We then select the Machine Learning workspace, and then we get started. Now, getting started. Azure Machine Learning takes us on a tour. Whether you prefer to write Python or R code, or zero code, low-code options such as the designer, you can build, train, and track highly accurate machine learning and deep learning models in an Azure Machine Learning workspace. Start training on your local machine and then scale out in the cloud. Assets will allow you to monitor your datasets and experiment, while managements will help help you manage your compute and data stores. You can pick your workspace; select your workspace and then move on. You can always view tutorials and documentation, and you can provide feedback. Congratulations! We just experienced Azure Machine Learning services.

Identifying and Sourcing the Relevant Data

Hi. In this lesson, we will identify and source the data relevant to our hypothesis. In sourcing and performing exploratory data analysis, the Flights Delays Data was sourced from the Azure Machine Learning datasets; however, it is included as an asset for you to be downloaded. This demo covers how to import data into PowerBI and how to perform our initial exploratory data analysis. You will look at how the data is structured and the columns present. Based on my hypothesis, we will also figure out which columns are of importance and subsequently, how they affect our model. We will then finish off with a sample report to get us on this journey. Now let's get to it. Here in our PowerBI desktop, we'll click on Get Data. We then go on to select Test or CSV since our data source is a CSV file. We select the Flights Delayed Data and then we open. PowerBI then connects to this data source and then loads the data into its memory. Once we explore the data and we're okay with it, we go ahead to load the data. PowerBI now fully loads this data into its memory. Here we see the columns involved: the year, month, day of the month, and then the day of the week. This tells us the exact dates a particular flight occurred. We have the carrier, which shows which particular airline was involved, the Oregon Airport ID and then the destination airport ID, showing us the starting and endpoints. We also have the deep delay, which signifies the departure delay in minutes, whether a flight was delayed or not, and then the minutes. So a negative number shows that the flight actually took place before the stipulated departure time. We also have the deep delayed 15, which shows whether a flight was delayed by 15 minutes; 0 for no and 1 for yes. We have a simple design dashboard to explore the data further. On the top right, you realize that we have the carrier, which is the airline which is involved, and out of these 2.7 million flights, we realize that 20% of these flights had departure delays, and 21% of these flights have arrival delays; 1% out of the 2.7 million flights were canceled. Which month has the highest departure delay? In this case, the sixth month. We should also drill down to see which day of the month experiences the highest delay in departure, which is the 10th day of each month. We can go down as low as seeing for each carrier and how each carrier is faring. On the bottom right corner, we also compared each carrier and how they are feeling when it comes to the delay in their departure and the delay in the arrival.

Training and Testing Data Subsets

Hi. In this clip, we'll be talking about the importance of splitting data into training and test subsets and some common methods of doing it. One of the first decisions to make when starting a modeling project is how to utilize existing data. One common method is to split the data into two groups, typically referred to as the training and testing sets. The training sets are used to develop models and feature sets. They are the substrate for estimating parameters, comparing models, and all of the other activities required to reach the final model. The test set is used only at the conclusion of these activities, for estimating the final unbiased assessment of the model's performance. It is critical that a test set not be used prior to this point. How should my data be set aside for testing? It is extremely difficult to make a uniform guideline. The proportion of data can be driven by many factors, including the size of the regional pool of samples and the total number of predictors. With a large pool of samples, the criticality of this decision is reduced once enough samples are included in the training sets. Also, in this case, alternatives to a simple initial split of the data might be a good idea. The ratio of the number of samples to the number of predictors is important to consider, too. We will have much more flexibility splitting the data when the number of samples is much greater than the number of predictors. However, when the number of samples is less than the number of predictors, then we can run into modeling difficulties. Even if the number of samples is similarly large. There are a number of ways to split the time into training and test sets. The most common approach is we use some version of random sampling. Completely random sampling is a straightforward strategy to implement and usually protects the process from being biased towards any characteristics of the data. However, this approach can be problematic when the response is not evenly distributed across the outcome. A less risky splitting strategy would be to use a stratified random sample based on the outcome. For classification models, this is accomplished by selecting samples at random within each class. Also, known random sampling can also be used when there is a good reason. One such case would be when there is an important temporal aspect of the data. Here it may be prudent to use the most recent data as it's tested. There are other data-splitting methods. For example, the trial-and-error method, which tries to overcome the high variance of the model performance by using simple random sampling by repeating the random sampling several times and then finding the average result. However, this is time consuming. Another method would be to use the systematic sampling, which is a deterministic approach designated for naturally ordered datasets such as time series. Also, there's a convenience sampling method, which is an efficient, deterministic method widely used when dealing with the time series. The dataset is split according to the discrete blocks. Example: the time intervals. The common and simplest way of splitting our data is by doing a 70% training set for training the model and 30% test set for validation.

Data Transformation Using Azure Machine Learning Components

For machine learning, data transformation entails some very general tags, such as joining datasets or changing column names. But it also includes many tags that are specific to machine learning, such as normalization, binning and grouping, and inference of missing values. In this lesson, we explore some models that are provided in Azure Machine Learning Designer for data transformation. Models for data transformation are grouped into the following tag base categories, creating filters for digital signal processing. Digital signal filters can be applied to numeric data to support machine learning tags, such as image recognition, voice recognition, and waveform form analysis. Generation and using count-based features. Count-based featurization models help you develop compact features to use in machine learning. General data manipulation and preparation. Merging datasets, cleaning missing values, grouping and summarizing data, changing column names and data types, or indicating which column is a label or a feature. Sampling and splitting datasets. Divided the data into training and tests, split the test sets by percentage or by filter condition, or perform sampling. Scaling and reducing data. Prepare numerical data for analysis by applying normalization or by scaling. Bin data into groups, remove or replace outliers, or perform principal component analysis. In this demo, we will be using Azure Machine Learning components in Designer to transform our data. Although we are using Azure Machine Learning Designer to do this data transformation, remember, data transformation can be achieved in any tool of your choice, whether you are using a Jupyter Notebook or a VS Code, you have more control when you're writing code as will be seen in later lessons in this course. Here in our Azure portal, we navigate to the Designer. Then we create a new designer project.. Let's take a look at the datasets available. There are a couple of datasets that Azure provides out of the box, and we are dealing with the flight data. So just drag and drop the dataset in our workspace. We can explore this data further by clicking on the dataset and then viewing the output. So this is exactly the data that was available to us when we did the exploratory data analysis in Power BI. Let's go ahead and close it. And then we want to know more about the data, so we use another ______ in Azure Designer, a Statistical Function, to summarize the data further. Now we link this to the flight data and the Summarize Data module. We click on the Summarize Data, but in order for us to summarize, we need to create a compute. So it's either we are using a default or we use another compute. I already have a compute set up, so I will use the default compute target. We go ahead to run this experiment to see how this data will be summarized. Now we have an error that says we should select compute target to run the pipeline. So now we select the compute that's available to us. Here I have the test compute that I created. I go ahead and save it and then run this again. We set up a pipeline for our run. We don't want to use an existing pipeline, so we create a new pipeline and call it dataTransformation, and then we click Run. So we go ahead to look at the output of the summarization. Then we visualize what we have in there. So the Summarize Data result visualization shows that there are 14 rows and 23 columns. So we can see the feature, which is the year, the day of the month, day of the week, carrier, OriginalAirportID, and so on. And on the next column you can see the Unique Value Count and then the Missing Value Count. We do not want to include missing values in our model, as it will end up skewing our model or giving us a wrong prediction. So there are some transformations that can be done to this data. Now, let's explore the data transformations available in Azure Machine Learning Designer. Remember, as I said earlier on, although we are using Azure Machine Learning Designer to do this, it can pretty much be done on your preferred machine learning environment using built-in Python SDKs or Azure APIs. So there a couple of data transformation modules. We can add columns, add rows, apply math operations, clean missing data, convert to CSV, edit metadata, join data, normalize data, and also we can split data. We can also select the columns in the datasets and also select the columns to transform. From our dataset available, which is the flight data, there are some columns which are categorical, and we need to let Azure Machine Learning know that these are categorical features. So, we go on to edit the metadata of these columns. Drag it into the workspace, and then join the data. There is some configuration that needs to be done. First of all, we need to select the columns we need to change the metadata for. Here we can select the columns by the names or by rules. Let's select the columns by the name. So, columns like the carrier, the origin airport, the destination airport, all are categorical features.

The Categorical we set to categorical features. The fields are also in unchanged so we leave them as is. So let's run the Edit Metadata module and see the outcome. Now, let's see the output. So, we can see that a column like the carrier has a categorical feature. We then want to remove columns with missing data, so we drag and drop the Select Columns module. And and then we select the columns to exclude. We can then run this to see the available columns now. Once we are done running this model, let's take a look at the output. We realize that the excluded columns are no longer visible. So these are some of the components that will be used when working on Azure Machine Learning Services.

Summary

So in this module, we introduced Azure Machine Learning Services and got it set up. We identified and saw the relevant data using the Azure Machine Learning datasets. We then performed exploratory data analysis using PowerBI. Then we distinguished the purpose of the training and test set, and also described some common methods of splitting data into training and test sets. We then performed data transformations using Azure Machine Learning Designer This has been fun, but there is more fun when we take a look at feature engineering and explore feature selection options in Azure Machine Learning, and also give a representation of the data we are dealing with. So, sit back and enjoy the next module.

Identifying Features from Raw Data

Overview

In this module, we will discuss the dependent and independent variables, which independent variables should we pick, and why. We will also discuss feature data conversion, table encoding, feature scaling. Hi, my name is Bismark Adomako, and welcome to the third model of our course, where we identify features from raw data. We'll be discussing some key concepts, such as selecting the level of data granularity appropriate to the hypothesis. We will also select the representation of the data appropriate to the family of models which will be employed. We will also be selecting features from the data in order to reduce the complexity in the data. Then we will apply common feature selection techniques and determine the features with the greatest impact on the model. In this module, we will be talking extensively about the data that we're going to use. This model will be more practical-based; hence, we will be having a lot of demos. Buckle up as we embark on this journey.

Selecting Data Granularity

It is important that we have the right level of data granularity for our hypothesis. So in this lesson, we'll be selecting the data granularity to use for our hypothesis. This lesson contains a demo on how Azure Machine Learning workspace is set up. The Azure Machine Learning workspace is a logical container for your machine learning experiments, compute targets, data stores, marching learning models, Docker images, and deployed services. It keeps them all together for teams to collaborate activity. We will be creating clusters in a Azure Databricks and also installing PyPI libraries in these clusters. We will select the level of data granularity, which we will use for our model. We'll also select a representation of the data, which is appropriate to the family of models, which will be employed. Now let's head over to our workspace. Now in our Azure Machine Learning portal, we create a dataset from the flights data we downloaded earlier. Select Datasets and then create a new dataset from our local machine. Click on Browse to locate the dataset, which is the Flight Delays Data, and then we open it. Once it's uploaded, we give the right name for the dataset. On the datasets type, we choose file. We can also use tabular. But for this purpose, we are using file. You can give a brief description of the datasets and then click Next. While the dataset uploads, as it's done uploading, we click on Create to create the datasets, which will now appear on our datasets page. Once created, we now get to the datasets. It gives us tags and sample usage, which we copy. Now, we've created an Azure Database cluster. This cluster will help us run our model. We give the cluster a name of flight_delays_cluster and then set autoscale to enable. We then terminate after 60 minutes of being idle. It creates the cluster. When the cluster is created, we add libraries to the cluster. Libraries are important because they allow us to bring dependencies into our model. In our case, we are using Azure AutoML. We are also using MLflow and also open datasets. So we'll install the appropriate libraries, which are necessary for this, which are necessary to run this notebook. Now in our notebook, we can create an Azure Machine Learning workspace. But we already have an Azure Machine Learning workspace, so we will not like to create a new one. Alright, I will use an the existing one. Here, we specify the subscription_id, the workspace_group, and in then a workspace_name. We run this and then we output the name of the workspace. We then create the tabular datasets from the Flight Delays Data. With the already set Flight Delays Data, we pick the file, and then we convert a file to a tabular form, which we then convert to a pandas Dataframe. We then describe the dataset to see what is in the datasets. As you can see, we have, as usual, the year, the month, day of the months up to the cancelled. You can see the number of count as well. Now let us perform basic statistical description of airlines. So we import Pandas and then set some display maximum rows, columns, and then the width. Once that is done, let is write a function, which will calculate the statistics by using simple groupby. As you can see, we have basic statistics about this. We have the count, the max, the min, and the mean. Now these are some of the things that you'll been doing on your own when you're running a model. So it's a fair idea that you have a good understanding of what is included here.

Dependent and Independent Variables

The two main variables in an experiment are the dependent and independent variables. An independent variable is a variable that is changed or controlled in a scientific experiment to test the effect on the dependent variable. A dependent variable is a variable being tested and measured. The dependent variable is dependent on the independent variable. As the experimenter changes their independent variable, the effect on the dependent variable is observed and recorded. For example, the scientists want to see if the brightness of the light has any effect on the moth being attracted to the light. The brightness of the lights is controlled by the scientists. This would be the independent variable. How the moth reacts to the different light levels, that's the distance to the light, would be the dependent variable. The independent and dependent variables may be viewed in terms of cause and effect. If the independent variable is changed, then an effect is seen in the dependent variable. Remember, the values of both variables may change in an experiment and are recorded. The difference is that the value of the independent variable is controlled by the experimenter while the value of the dependent variable only changes in response to the independent variable. To help us remember and differentiate clearly between the dependent and independent variable, when results are plotted in the graphs, the convention is to use the independent variable as the X axis and the dependent variable as the Y axis. That DRYMIX acronym can help keep the variable straight. The DRY acronym: D is a dependent variable, R is a responding variable, Y is the axis on which the dependent or responding variable is graphed, in this case, the Y axis. MIX: M is a manipulated variable or the one that is changing an experiment. I is the independent variable. X is the axis on which the independent or manipulated variable is graphed, here, the horizontal axis. So back to our datasets, it can be seen clearly that the year, month, day of the month, day of the week, carrier, origin airport ID, destination airport ID, CRS departure time, CRS arrival time, and then cancelled are independent variables. This is because while we change any of these variables, we realize the effects on the arrival delay. So in this case, the arrival delay 15 is the dependent variable. So in machine learning, you may think of the independent variables as the features and the dependent variables as the labels.

Choosing the Right Encoding Method

In machine learning models, we are often required to convert the test features to its numeric representation. The two most common ways to do this is to use the label encoder or one-hot encoder. The accuracy of the model may shift by large numbers by using the right encoding and the right scenario. Now, let us understand the working of the label and one-hot encoder. And further, you will see how to use these encoders in our model and see the impact on predictions. Label encoder encode labels with a value between 0 and n_classes-1 where n is the number of distinct labels. If a label repeats, it assigns the same value to as assigned earlier. One-hot encoding takes a column, which has categorical data, which has been labeled encoded and then splits the column into multiple columns. Consider the below example. If you have to pass this data to the model, we need to encode the Country column to its numeric representation by using a label encoder. After applying the label encoder, you will get a result as seen on the right. That's all label encoding is about, but depending on the data, label encoding introduces a new problem. For example, we have encoded a set of country names into numerical data. This is actually categorical data, and there is no relationship of any kind between the rows. The problem here is, since they are different numbers in the same column, the model will misunderstand the data to be in some kind of order, 0 is greater than 1, 1 is greater than 2. The model may derive a correlation like as the country number increases, the population increases. But this clearly may not be the scenario in some other data or the prediction set. To overcome this problem, we use the one-hot encoder. Now, as we already discussed, depending on the detail we have, we might run into situations where, after label encoding, you might confuse our model into thinking that a column has data with some kind of order or hierarchy when we clearly don't have it. To avoid this, we one-hot encode that column. What one-hot encoding does is it takes a column, which has categorical data, which has been label encoded, and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has hot value. In our example, you'll get four new columns, one for each country, Japan, US, India and China. For rows which have the first column value as Japan, the Japan column will have a 1 and then the other 3 columns will have 0s. Similarly, for rows which have the first column value as the US, the US column will have a 1, and the other 3 columns will have 0s, and so on.

Feature Scaling

Feature scaling can vary your result a lot while using certain algorithms and have a minimal or no effect on others. To understand this, let's look at why features need to be scaled, varieties of scaling methods, and when we should scale our features. Most of the times, your datasets will contain features highly varying in magnitudes, units, and range, but since most of the machine learning algorithms use Euclidean distance between two datapoints in their computations, this is a problem. If left alone, these algorithms only take into the magnitude of features, neglecting the units. The results will vary greatly between different units. Example: 5kg and 5000g would differ greatly. The features with high magnitudes will weigh in a lot more in the distant calculations than features with low magnitudes. To suppress this effect, we need to bring all features to the same level of magnitudes. This can be achieved by scaling. So how do we scale features? There are four common methods to perform feature scaling. Standardization replaces the values by their Z-scores. Mean normalization distribution will have values between -1 and 1, with new equals 0. Standardization and mean normalization can be used for algorithms that assume zero-centric data like principal component analysis. Min-max scaling: This scaling brings the value between 0 and 1, while in unit scaling, scaling is done considering the whole feature vector to be of unit length. Min-max scaling and unit vector techniques produces values of ranges 0 and 1. When dealing with features with hard boundaries, this is quite useful. For example, when dealing with image data, the colors can range from only 0 to 255. So the bigger question is, when do we scale? The rule of thumb here is an algorithm that computes distance or assumes normality, scaled features. For example, algorithms where feature scaling matters. Ah! K-nearest neighbors, Principal Component Analysis, and some tree-based models. Algorithms like Linear Discriminant Analysis and Naive Bayes are by design equipped to handle this and gives weight to the features accordingly. Performing feature scaling in these may not have much effect.

Common Feature Selection Techniques

In this lesson, we discuss some common feature selection techniques and also why it is important to do this. Before we proceed, we need to answer this question. Why don't you give all the features to the machine learning algorithm and let it decide which feature is important? So there are three reasons why we don't, the curse of dimensionality and not always overfitting. If you have more columns in the data than the number of rows, we will be able to fit our training data perfectly, but that won't generalize to the new samples, and thus, we learn absolutely nothing. We want our models to be simple and explainable. We lose explainability when we have a lot of features. Most of the times you will have many non-informative features, for example, name or ID variables. Poor quality input will produce poor quality output. Also, a large number of features make a model bulky, time taking, and hard to implement in production. There are a lot of ways in which we can think of feature selection, but most feature selection methods can be divided into three major buckets: filter-based, where we specify some metric and base on that filter ______ features. An example of such a metric could be correlation or chi-square. Wrapper-based. Wrapper methods consider a selection of a set of features as a session problem. Example will be the recursive feature elimination. Embedded methods use algorithms that have built-in feature selection methods, for example, Lasso and RF have their own feature selection methods. In this demo, we apply common feature selection techniques to determine features with the greatest impact on our model. You also select features from the data in order to reduce the complexity of the data using Azure machine learning services from our previously created Databricks notebook. At this stage, let us examine how complete the data is. We realize that the variables ArrDelay, DepDeplay, and DepDel15 all have missing values. Hence, we don't want to skew our model, so you remove the null values from these columns, meaning that the total account of rows is now going to decrease. But it's for the good because, as I said early on, we do not want to skew our model with null values. So we run the dropna function in a data frame. We should drop all null values. Let us now calculate missing values again, and this time we realize that there are no missing values. You can take a look at the data frame as it stands now; however, we realize that DepDel15 is a float instead of an integer or a categorical feature. Mind you, DepDel15 is actually the label we are trying to find. So, let us go ahead and then convert it into an integer. Once that's done, let us again look at the data frame. In this case, we realize that DepDel15 is now an integer and no longer a float. Now, let us ______ columns that are possible target ______, the DepDel, the ArrDel, Cancelled, and then Year. This is because they are well correlated with the label we are trying to find. It is understandable that if a plane arrives late, it's quite possible that it's going to depart late. And even if a flight was cancelled, then it is highly due to the fact that it arrived late or the departure time was exceeded. So, let us drop this column before we continue with our experiment. Once that is done, let us take a look at the data frame to see if this column still appears. Great. We do now have these columns appearing. Now, we want to optimize our data frame, hence, we create a Spark DataFrame from a pandas DataFrame using an arrow. So running this code, we now convert the pandas DataFrame into a Spark DataFrame. Remember we talked about one hot encoding and feature scaling. Great. This is where it's going to be of importance. For simplicity sake, we will use one hot encoding to convert all categorical variables into binary vectors. It is possible here to improve prediction accuracy by converting each categorical column with an appropriate method. Here, we will use a combination of StringIndexer and OneHotEncoderEstimator to convey the categorical features. The OneHotEncoderEstimator will retain a SparseVector. Since we have more than one stage of feature transformation, we will employ a pipeline to tie the stages together. This will make our code simpler. So it lets you import the necessary models to use here. We also define the categoricalColumns. Now we convert our categorical columns appropriately into categorical forms. And then add this stage to the stage of our pipeline. The above could basically index each categorical column using the StringIndexer and then converting indexed categories into one hot encoded variable. The resulting output has the binary vectors appended to the end of each row. We use the StringIndexer again to encode our labels to label indices. Let us convert the label into label indices using the StringIndexer, and then we add it as a stage to our pipeline. Now, we use a VectorAssembler to combine all the feature columns into a single vector column. This includes both the numeric columns and then the one hot encoded binary vector columns in our data sets. Here, you transform all features into a vector using a VectorAssembler. And then we add it as a stage in our pipeline. We now run the stage as a pipeline. This puts the data through all of the feature transformations we described in a single call. Now, let us take a look at our prepared data. Nice. We see all the columns that we are using for our experiment, including the index column and also the vectorized columns. As a final step in this experiment, we will not need all the columns since we have been able to index all the columns and also vectorized them. Now, we are going to select the features and the labels and then a third column, which should be the month. The month will come in useful when you want to split the data. So, our final data set contains the level, the features, and then the month. We are basically going to use our levels and features in our training and in testing. The month that was included here will help us split the data accordingly. You know, on this model, we'll discuss why using a random split will be of less importance.

Summary

In this module, we identified of the data granularity appropriate to the hypothesis by selecting a representation of the data, we discussed extensively dependent and independent variables. We also talked about encoding methods and why it is important to encode your features. Then we talked about feature scaling. We ended up the feature selection where we talked about common feature selection techniques, and then how to reduce the data complexity. Stay tuned for our next model, where we talk extensively about building the model, creating experiment, and then submitting this experiment to Azure Machine Learning. We then track and monitor how our experiment is doing.

Building the Model

Overview

We are finally going to build a model of our data. Hi, my name is Bismark Adomako, and welcome to this module of our course where we build the model. So, in this module we will discuss which models to choose and why. We will discuss various model evaluation techniques for the data we have supplied. Once a ______ of the model is obtained, we will discuss how we can optimize the model and retest. We will looking at critical concepts, such as selecting the family of models appropriate the data and the hypothesis. We will look at some classification models and the consideration when choosing an algorithm. We will also look at Azure Machine Learning classification model options. And then we will identify metrics of a model used to determine how well it fits the dataset we have. We will also be evaluating the model for accuracy and fit to our dataset. We then identify common methods of model validation, including cross-validation and ROC curve analysis. We finally discuss hyperparameter optimization strategies using Azure Machine Learning services.

Selecting the Appropriate Model

In this clip, we'll be selecting the appropriate model for our hypothesis. There are several specific types of supervised learning that are represented within Azure Machine Learning service: classification, regression, and anomaly detection. Classification: when the data is being used to predict a category. Supervised learning is also called Classification. This is a case when you want to tell whether a flight will be delayed or not. Classification answers the question, what's class, and applied when the output has finite and described values. Regression, on the other hand, is when a value is being predicted, as with stock prices. Supervised learning here is called Regression. It answers the question, how much and supplied when the output is a continuous number. Now let's talk a little bit about classification, since our hypothesis here is a classic car classification problem. Classification specifies a class to which data elements belong to and is best used when the output has finite and discrete values. This we already talked about. It predicts a class when input is variable as well. When there are only two choices, it is called two-class, or binomial classification. When there are more categories to predict, it is known as a multi-class classification. Now will a flight be delayed more than 15 minutes? Is it a two-class classification or a multi-class classification? This is straightforward and obvious enough. Classification models can be linear. For example, logistic regression and support vector machines. They can also be non-linear models. Example: Naive Bayes, decision tree, and random forest. There are some key considerations to make when choosing an algorithm for our model prediction. Getting the most accurate answer possible isn't always necessary. Sometimes ______ is adequate, depending on what you want to use it for. If that's the case, you may be able to cut your processing time dramatically by sticking with more approximate methods. Training time; the number of minutes or hours necessary to train a model varies a great deal between algorithms. Training time is often closely tied to accuracy. One typically accompanies the other. In addition, some algorithms are more sensitive to the number of data points than the others. Linearity: lots of machine learning algorithms make use of linearity. Linear classification algorithms assume that classes can be separated by a straight line. These include logistic regression and support vector machines as implemented in Azure Machine Learning services. The number of parameters: Parameters are the ______ a data scientist gets to tend when setting up an algorithm, the numbers that affect the algorithm's behavior, such as error tolerance or number of iterations or options between variants or how the algorithm behaves. The training time and accuracy of the algorithm can sometimes be quite sensitive of getting just the right settings. Typically, algorithms with large numbers of parameters require the most trial and error to find a good combination. Alternatively, there's a parameter stripping model block in the classic version of Azure Machine Learning's designer that automatically tries all parameter combinations at whatever granularity you choose. The number of figures also plays an important role. For certain types of data, the number of features can be very large compared to the number of data points. This is often the case with genetics or test raw data. The large number of features can bog down some learning algorithms, making training time unfeasibly long. Support vector machines are particularly well suited for this case, because they are very fast. Now let's talk a little more about the algorithm implemented in this scenario, the logistic regression. Although it includes regression in the name, logistic regression is actually a powerful tool for two-class and multi-class classification. It's fast and simple. The fact that it uses on S-shaped curve instead of a straight line makes it a natural fit for dividing data into groups. So when building our model, we use a logistic regression.

Azure ML Classification Model Options

In this clip, we are talking about Azure Machine Learning classification option. This lesson contains a demo where we explore the Microsoft Azure Machine Algorithm Cheat Sheet. The purpose of the cheat sheet, which is also included in this model, is to help us make decisions when we are dealing with machine learning on Azure Machine Learning Services. We will then build our model and then talk about the classification models that we employed. But first, let us explore the algorithm cheat sheet. In this cheat sheet, we realize that we start with a question what we want to do. Do we want to extract information from test? Do we want to predict values? Do we want to generate recommendations? Do we want to predict between several categories? Or do we want to predict between two categories? In our case, it is exactly what we want to do. So it is a two-class classification which answers simple two-class questions like yes or no, true or false. Will it be delayed or not? Some of the algorithms present in Azure Machine Learning Services are Two-Class Support Vector Machines, which do under 100 features, linear model. Two-Class Averaged Perceptron is a fast-training linear model. Two-Class Decision Forest is accurate and fast training. Two-Class Logistic Regression is fast training and is a linear model. You also have Two-Class Boosted Decision Tree and Two-Class Neural Network. Now, we know exactly what we are going to do and how to do it, so let's get to the demo, where we talk more about building in our model. So far we've worked with our data from Azure Databricks. What we did in Azure Databricks was quite interesting, and it gave us a chance to explore the data further. We had full control of our data. We were able to perform one-hot encoding and also feature scaling. That is the power that Azure Machine Learning brings to you. We could actually complete the whole machine learning design process from Databricks, but Azure Machine Learning itself comes with a designer which looks like the classical Azure Machine Learning Studio that we already know. We are going to use the designer in building our model. It's going to be fun, and it's going to be fast, and it's going to be easy. But before we do that, let's create an Azure Machine Learning compute by clicking on Compute. We navigate to the Training Clusters, and then we add a new one. So here we specify the compute name. Then the region is already specified because the region is tied to the machine learning workspace. We select the virtual machine size. A Standard_D12_v2 to with 4 VCPUs, 28 GB RAM, and then 200 GB resource disk will be enough for this experiment. We then specify the number of minimum nodes we want, and then the maximum number of nodes. We then click on Create, but, as you can see, I already have a compute created for this experiment. Once we've created, we can see the compute just created FlightDelays compute. There are some details about the cluster, the allocation state, the allocation state transition time, the provisioning state, and then the created date. Since we specified the minimum number of notes to be 2, at every point in time, there will be 2 compute nodes which will be available to us. But then if it needs more compute, it will scale out. So that's that with creating our computes. Now, let us head over Azure Machine Learning Designer and then create our experiments. So we click on Designer, and then we create a new designer pipeline. On the right you can see the settings of this pipeline. Let's ______ to select the compute target we created. So we select the FlightDelays compute so that it persists throughout the experiment. Now I specify the name of the designer experiment. We can add a little bit of description about our experiment.

So on left pane, we realize that we have datasets that are available. We can see the dataset we uploaded earlier, the Flight Delays Data. We can see other datasets as well. Out of the box, Azure Machine Learning provides sample datasets. Let's locate the Flight Delays Data and then drag it onto our workspace. We can take a look at the datasets we have and a type of data for each column. The year here is a numeric feature. The month is also a numeric feature. We can also see that Carrier is a string feature with no missing values. The Origin ID Is a numeric feature; the destination ID is also a numeric feature. But this shouldn't be the case because the Carrier, OriginAirportID, DestAirportID are supposed to be categorical features, so we edit the features as such. And to help us do this, we use the Edit Metadata model, which can be found under the Data Transformation pane. Once we drag it onto our workspace, we link our dataset to it. We then click on it and then select the columns we want to edit types for. Let's select the columns by name. The Carrier, OriginAirportID, and then the DestAirportID. Then we click on Save. The data type is unchanged, but then we set Categorical to Categorical. Once we are done with editing metadata for our columns, let us run what we've done so far. We go ahead and then create a new experiment, and then we run. The experiment will now be submitted to our compute to be run. Remember in our Azure Databricks notebook, we ______ columns which are highly correlated to the label we are trying to predict. There's a module to do that in Azure Designer. Let's use the Select Columns in Dataset model. We then connect the output of the Edit Metadata module to the input on the Select Columns in Dataset. Now, we click on the Select Columns in Dataset and then select the columns to exclude. Now we include all the columns and add an additional step, which will exclude just the year and the departure delay in minutes. We save, and then we run our step. Once our Select Columns in Dataset model is done running, let's take a look at the data columns available. We do so by clicking on the module and then navigate to Outputs. We then visualize the results from our dataset. First of all, we want to check if the year is still available. The year is still available, so is the delay in the departure in minutes. Now, let's take a look at the wholeness of this data by clicking on the column and seeing statistics about the column. We can see that for the month we have 7 unique values with no missing values. The day of the month also do not have any missing values. Day of the week do not have any missing values. The departure delay. 15 have missing values. They arrival delay also have missing values. So we have to take time and then treat them as such. So to help us take care of missing values, let us use the Clean Missing Data model under our Data Transformation pane. We drag and drop it onto our workspace, and then we connect to the output of the Select Columns in Dataset to the input of the Clean Missing Data model. We click on it and then select the columns to be cleaned. So we include the columns we want to clean. The DepDel15 has missing values, and then ArrDelay also have missing values. We then save and run our model so far. Once the cleaning is completed, let's take a look at the dataset once more by clicking on the module and then navigating to the Outputs pane. Now let's look at the clean data, with particular attention to the DepDel15. We realize that there are no missing values. And also for ArrDelay there are no missing values. Now, let us close this and then split our data. Remember we talked about techniques of splitting data, whether you split randomly or using the stratified split. We also talked about splitting data based on special scenarios. Well, we are using one special scenario over here. Because we are going to do predictions, and in our dataset, we have data for 2013, we would like to learn from past months and then predict on the present month. We are going to split our data into two using the month. We will use the months which are less than 10, that is October, as the training set, and the month greater than or equal to 10 as the testing set. So to help us do that, still under data transformation pane, we will use the Split Data. Now, we drag the clean data outputs of the Clean Missing Data model to the input of the Split Data model. Let us do the configurations for our Split Data model. Splitting mode, are we just splitting? No. ______ Relative Expression split. Over here we specify the column we want to split on, which is the month. And we want to split for all values less than 10. We do this, and then we run our model step. Now that our data is split, it is now time to build the model. Remember we talked about classification types in Azure Machine Learning? We are going to use a two-class classification algorithm to train our model. So we navigate to the Machine Learning Algorithms pane, and then since our problem is a classification problem, we select a Two-Class Logistic Regression. Drag it and drop it on our workspace. We are going to predict a single parameter. For now, leave the rest of the configurations as they are.

Metrics for Model Classification and Evaluation

There are some metrics that you like to report when evaluating classification models. If you compare models, they are run by the metric you select for evaluation. The accuracy measures a goodness of a classification model as a proportion of true results throughout our cases. The precision is a proportion of true results over all positive results. The recall is a fraction of all correct results returned by the model. The F-score is computed as a weighted average of precision and recall between 0 and 1 where the ideal F-score value is 1. Now the AUC measures the area under the curve plotted with true positive on the Y axis and false positive on the X axis. This metric is useful because it provides a single number that lets you compare models of different types. The average log loss is a single score used to express the penalty for wrong results. It is calculated as a difference between two probability distributions, the true one and the one in the model. Finally, the training log loss is a single score that represents the advantage of the classifier over a random prediction. The log loss measures the uncertainty of your model by comparing the probabilities it outputs to the unknown variables that is a ground truth in the levels. Remember, you want to minimize log loss for the model as a whole. These metrics for model classifications will be seen when we explore experiment from the Azure machine learning service. So far, we've talked about how to create pipelines, how to create experiments, and how to submit these experiments to our remote compute that already exists. We also talked about how to train the model using logistic regression. Now, let us understand fully how model evaluation works. You covered demo topics, such as generating scores on our model and using common methods of model validation. We then compared scores for two different models and then we optimize our hyperparameters to see how well the model will still perform. Once we have our model, we have to train the model. We do so by navigating to the Model Training pane and then selecting Train Model. We drag and drop the Train Model onto our workspace, and then we connect the outputs of the two-class regression model to the inputs of the untrained model. We also connected the result set of this dataset1 to the dataset's inputs of the train model module. Now, let's configure our training parameters. The label column here is what you are trying to predict, and the column name is Departure Delay 15. Remember, you are trying to predict through it a flight will be delayed by 15 minutes. Then we save. We'll then need to score our model when it is done training. So we now get to the Model Scoring and Evaluation pane and then drag and drop the Score Model. The Score Model takes the train model as an input and then the test dataset has dataset input. Once we've scored our model, the next thing you'd like to do is revaluate our model. So we drag and drop the Evaluate Model module onto our workspace. Then we connect the output of the Score Model to the input of the Evaluate Model. The Evaluate Model will help us evaluate the metrics of the module, so things like the ROC, the confusion matrix, and then the accuracy will be ______ for by the Evaluate Model module. Now let us go ahead and then run it. Once you are done training the model, you can see the output of the train model and then save the trained model. WE can also download the trained model. We can also see the score of the model by navigating to the Outputs pane and then visualizing the result. Now you can see all the columns we use in our training with additional column, the scored label, and then the scored probabilities. The trained model assigns probability to each prediction in mix. Now, let us talk about the evolution of the model. When you navigate to the output of the evaluation model module and then visualize the evaluation model. First, the ROC curve. ROC stands for receive operating characteristics, and is a plot of the correctly classified levels versus the incorrectly classified levels for a particular model. We see that we have a pretty good ROC curve. Precision-recall chart. With this chart, you can compare the precision recall curves for each model to determine which model has an acceptable relationship between precision and recall for a particular business problem. The term precision represents that ability for a classifier to label all instances correctly. Recall represents ability for a classifier to find all instances of a particular level. Ideally, the model would have 100% precision and 100% accuracy. Now, the lift curve. You can compare the lift of the model build automatically with Azure machine learning to the baseline in order to view the value gain of that particular model. Lift curves are used to evaluate the performance of a classification model, it shows how much better you can expect to do with a model compared to without a model. Lastly, the confusion matrix. The confusion matrix is used to describe the performance of a classification model. Each row represents the instance of the true class and each column represent the instance of the predicted class. The confusion matrix shows the correctly classified levels and the incorrectly classified levels for a given model. For classification problems, Azure machine learning automatically provides the confusion matrix for each model that is viewed. With that kind of color, the higher count in the particular part of the matrix, ideally, the darkest colors will be along the diagonal of the matrix. Now we have the accuracy, the precision-recall, F-one score, and AUC. You realize that this model has a very high accuracy of 92%, and this is a very well-performing module.

Summary

We started this module by selecting the appropriate model for the hypothesis where we talked about the classification models and the considerations when choosing an algorithm. Then, we introduced Azure machine learning classification model options. And then we identified the metrics for model classification. We talked extensively about training the model and how the model evaluation works. It has been a long journey so far, and I hope it was worth it. And on this module, we'll be talking more about monitoring of our model and how to tell your model is drifting. Don't worry if you do not understand the keyword drifting. We will be introducing them in our next module. Stay tuned and enjoy the lesson.

Monitoring and Managing the Performance of a Model

Overview

Hi, my name is Bismark Adomako, and welcome to this model of our course where we monitor and manage the performance or our model. In this model, we will discuss how we can put the model into production. Once in production, how it can be monitored and retrained if and when necessary. We will look at some key concepts, such as differentiating model predictions and residuals, identifying reasons why model performance may degrade over time, identifying metrics used to evaluate a model's performance, describing common tactics for model retraining, and lastly, we will design a model performance management strategy.

Deploying Model as Batch Inference

In this clip, we talk about how to deploy a model as a batch inference pipeline. This is basically a demo describing how batch inference pipelines work and the strategies for model retraining. Our pipeline is done running, and we talked about the Azure machine learning's core model, the scored labels, and the probabilities. We also talked about the evaluate model where we look that the ROC curve, the position-recall curve, and also the confusion matrix. You also looked at the accuracy, the precision, the record, the F-one score, and also the AUC. Now, once you are done running our model, the next thing to do is to publish it as a pipeline. Now let us go ahead and then create a new pipeline. You can then view the publisher's pipeline details, and then run this pipeline when you want to. We also have run details of the pipeline. Currently, the pipeline does not run yet. Now back at our designer interface. Once we create an experiment and a pipeline, most of the things that we want to do will be constantly predicting values. It can either be a log data set or real-time prediction, and to do this, next to the Run button, we have Create inference pipeline. We can either create a real-time inference pipeline or a batch inference pipeline. Now let us learn how to use the designer to set up a batch prediction pipeline and then a web service. The batch prediction allows for continuous on-demand scoring of training models on live data sets, optionally configured as a web service that can be triggered from any HTTP library. Let us click the dropdown and then select Batch inference pipeline. The result set default batch inference pipeline. This includes a node for your original pipelines permit set up, a node for raw data for scoring, and a node to score the raw data against your original pipeline. You can add other nodes to change the behavior of the batching for its process. For example, we can create a partition and sample node and place it between the raw data and in the scoring nodes. Now we are ready to deploy the pipeline. Click the Publish button, which opens up the interface for us to set up our endpoints. Click the dropdown and select a new pipeline. We give the endpoint a name and an optional description and publish it. After the deployment is complete, let us go to the Endpoints tab and then click the name of the endpoints we just created. This is a pipeline endpoint. The pipeline endpoint is FlightDealyPrediction-batch inference pipeline. The Pipeline Details page shows you the detailed information about the pipeline, the status, the REST endpoint for you to call the pipeline, who published it, the date it was published, the endpoint ID. Now let us see the published pipeline. The Pipeline Details Page shows the detailed run history and connection string information for your pipeline. Click the Run button to create a monitored run for the pipeline. In the run setup, you can provide an experiment name and then a compository run description. You can also change the value of any pipeline params you had parameters set up. You then click on Run to run the pipeline. Now it shows up there that the pipeline has been submitted. You can view details about this pipeline Run. The Consume tab contains the REST endpoint for rerunning your pipeline. To make a REST call, you need an OAuth 2.0 bearer type authentication header. On the Consume page, you see the basic information about pipeline, the REST endpoints, and then the consumption option to be used in C#, Python, or R.

Deploying Mode as Real-time Inference End Points

Before we talked about how to deploy a model as a batch inference pipeline. Now, let us take a look at how to deploy a model as a real-time inference pipeline where we can make API calls to its endpoint. This clip is basically a demo demonstrating how batch inference pipelines work and how to make API calls to real-time endpoints. Since you like to submit API calls to our real-time engine to make real-time predictions, to do this, let us navigate to the publish Pipelines and then over to the Pipeline drafts. We can see the FlightDelaysPrediction pipeline we published earlier on. Let's click on that. While this is up, we can update an inference pipeline we already created, or we can create a new inference pipeline. We want to create a real-time pipeline, but before we do this, we need an Azure community service compute. We already talked about how to create computes. Now, let's create a compute, which is an Azure communities type. To do this, you navigate to the compute and also navigate to the inference computes. You can see that I already have an inference compute created, but let me walk you through the steps of creating an inferenece compute. First, you give it a name and then you select the region. We then select the virtual machine size and then specify the cluster purpose, whether it's for production or it's for Dev-test. Now we want to move our model in production, so we keep it at production. We then specify the number of nodes here. We specify 2. We then click on Create to create the Kubernetes Service. I'll be creating a new one since I already have a Kubernetes service. Now, let us go back to our pipeline and then create a real-time inference pipeline. So with a real-time inference pipeline, we have additional nodes which are attached to our pipeline, for example, the Web Service Input and also the Web Service Output. In order to deploy this real-time pipeline, we have to run it. So let us go ahead and run this inference pipe line. You select an experiment and then use the default compute target we created, FlightDelays, to run this pipeline. Once our real-time inference pipeline is done running, it's now time to deploy. We do so by giving it a name and then an optional description, and then we select the Azure Kubernetes service that we created, and then deploy. And now it shows that it's preparing to deploy. Now that our real-time pipeline succeeded, deploy, let us view the real-time endpoint. The details give us extensive details about deployed real-time endpoint. We have the deployment state, the service ID, the REST endpoint, and whether you are using the key-based authentication. You also have the memory, the scoring timeout, and whether Autoscale is enabled. Let's now get to the Test tab and test our real- time endpoints. So we are going to make predictions to the real-time endpoint. Now, we want to predict for a month in October. So let us test. This gives us this code label and then the scored probability as well, meaning that on the 19th day of October 2018 when there's a flight with a carrier DL and then from this origin airport and to this destination, the flight is not going to be late. And then we also have the calculated probability. These are ways that we can test our real-time endpoint. There's also provisions to consume our real-time endpoint programmatically by making calls to our REST endpoint either using a key or a token. Here, we have some consumption options in C#, Python, and in R. These consumption options can be used to make real-time predictions by specifying the data input and giving the required amount of details. So that is it. We've been able to predict if a flight will be delayed more than 15 minutes with the score levels and then the score probabilities. While our model is in production, you like to monitor how our model is doing from time to time and check for data drift.

Model Performance Strategy

Hi. In this clip, we talk about how to design model performance management strategy. But before that, let us introduce the concept of data drift. You may be wondering what data drift is. In the context of machine learning, data drift is the change in model input data that leads to model performance degradation. It is one of the top reasons why model accuracy degrades over time. Thus, monitoring data drift helps detect model performance issues. But the question is, why does a model's performance degrade over time? The simple answer is source data for big data applications is subject to constant and unexpected change. So what can we monitor in Azure Machine Learning? With Azure Machine Learning, you can monitor the inputs to a model deployed on Azure Kubernetes Service and compare this data to the training datasets for the model. At regular intervals, the inference data is snapshotted and profiled, then computed against the baseline datasets to produce a data drift analysis that measures the magnitude of data drift called a drift coefficient, measures the data drift contributions by feature, indicating which features cause data drift. It also measures distance metrics. Currently Wasserstein and energy distance are computed. It also measures the distribution of features, currently kernel density estimation in histograms. Then it sends alerts to data drift by email. You may be wondering how data drift is monitored in Azure Machine Learning. So using Azure Machine Learning, data drift is monitored through datasets or deployments. To monitor for data drift, a baseline dataset, usually the training dataset for a model, is specified. A second dataset, usually model inputs data gathered from a deployment, is tested against the baseline dataset. Both datasets are profiled and input to the data drift monitoring service. A machine learning model is trained to detect the difference between the two datasets. The model's performance is converted to the drift coefficient, which measures the magnitude of drift between the two datasets. Using model interpretability, the features that contribute to the drift coefficient are computed from the dataset profile. Statistical information about each feature is also tracked. Now to retrain your model after data drift, when data drift negatively impacts the performance of your deployed model, it is time to train the model. To do so, proceed with the following steps. Investigate the collected data and prepare data to train the new model. Split it into training and test sets. Then, train the model again using the new data. Evaluate performance of the new generated model. And finally, deploy the new model if the performance is better than the production model. Incorporating a new state-of-the-art machine learning model into a production application is a rewarding yet often frustrating experience. Here we provided tips for designing a model performance management strategy. First of all, you centralize your data pipelines. In ______, datasets are provided, but in production, managing, securing, and wrangling data is much more challenging. Before putting the ______ of the ______ model in production, it's important to understand your data and use case. Services, such as Azure Storage, Azure Event Hub, and Azure Data Explorer enables ______ ingestion and processing of your data. The Azure Machine Learning service directly integrates with these services through the MLOps SDK. As your team grows and your projects become bigger, you will need to keep track of your team's work. Now, in ______ settings, it is easy to get used to working on the model alone. But in production, keeping track of your work done across distributed teams can be challenging. Luckily, the Azure Machine Learning workspace enables teams to manage experiments and keep track of their work. Also, we need to deploy a solid baseline. Use low-priority compute and autoscaling. When possible, deploy to the edge and leverage distributed training. Before deploying a complex state-of-the-art model into production, it makes sense to deploy a solid baseline model. Having a solid baseline model provide sufficient advantages. A strong baseline model helps to better understand the deployment process for getting a model into production, evaluate impact of a given solution, provide a point of comparison. The Azure AutoML service helps generate a strong baseline for many tags without the need to even write a single line of ML code. AutoML automates the process of selecting the best algorithm to use for your specific data, so you can generate machine learning baselines quickly. Now to use low-priority compute and autoscaling, training large state-of-the-art models can be computationally expensive, however taking advantage of how priority compute can lead to significant savings by an order of magnitude. Then, we leverage distributed training. There are two traditional approaches to distributed training, distributed batching and distributed hyperparameter optimization. Without distributed training, it would be difficult to scale the state of model to production performance. Additionally, the Azure ML service provides hyperdrive, a hyperparameter optimization service that uses state-of-the-art methods for hyperparameter tuning. Also, we have to perform AB testing of different models. The Azure ML service makes it easy to manage versions and, if necessary, roll back models. While it may seem that a model performs better in the training environment, in production, you may see a different result. An AB test enables you to test different model versions across different audiences based on conversions, KPIs, or other metrics. You can decide which one performs best or if it's time to iterate the next version of the model. Finally, we need to monitor for data drift and retrain the model if necessary. We already talked about how to retrain the model, and we already know that data drift happens when data served through a model in production is different from the data used to train the model. It is one of the top reasons where model accuracy decreases over time. Thus, monitoring data drift helps detect model performance issues.

Summary

In this model, we differentiated model predictions and residuals. We identified reasons why model performance may degrade over time, popularly known as model drifting. We also identified metrics used to evaluate the model's performance. Then we described common tactics for model retraining, and finally, we designed a model performance management strategy. It has been a long journey so far: creating a hypothesis, sourcing and transforming data, identifying features from raw data, building a model of our data, and finally monitoring and managing the performance of our model. In doing all this, we answered the ultimate question, will a flight be delayed more than 15 minutes, by testing our deployed model with a real-time inference pipeline. Imagine predicting if your flight will be delayed under pressure or however, and by how many minutes or even the worst case, whether your flight will be canceled. This same process we encountered can be used to solve such problems. So go ahead and enjoy building models in Azure Machine Learning services.

Course author

Bismark Adomako

Bismark believes that education is no longer a one-time investment, but instead a lifelong pursuit and that great authors and mentors can play an invaluable role in finding and developing a...

Course info

Level	Advanced
Rating	★★★★★
My rating	★★★★★
Duration	1h 40m
Released	13 Dec 2019

Share course