ステップ 2　統計的学習フレームワークを利用し学習モデルを作る

●Azure ML 上で学習モデルを生成

（さて、先ほど作成した`trainingdata.csv`を学習データとして学習モデルを作成します。）

https://portal.azure.com/ にアクセスし Azure ポータルにログインします。左メニューから[参照 >]を選択し、フィルターに"machine"と入力すると一覧に[Machine Learning ワークスペース]が表示されます。この☆マークをオンにするとメニューに登録されます。これをクリックして Azure ML のホーム画面を表示します(図 4)。

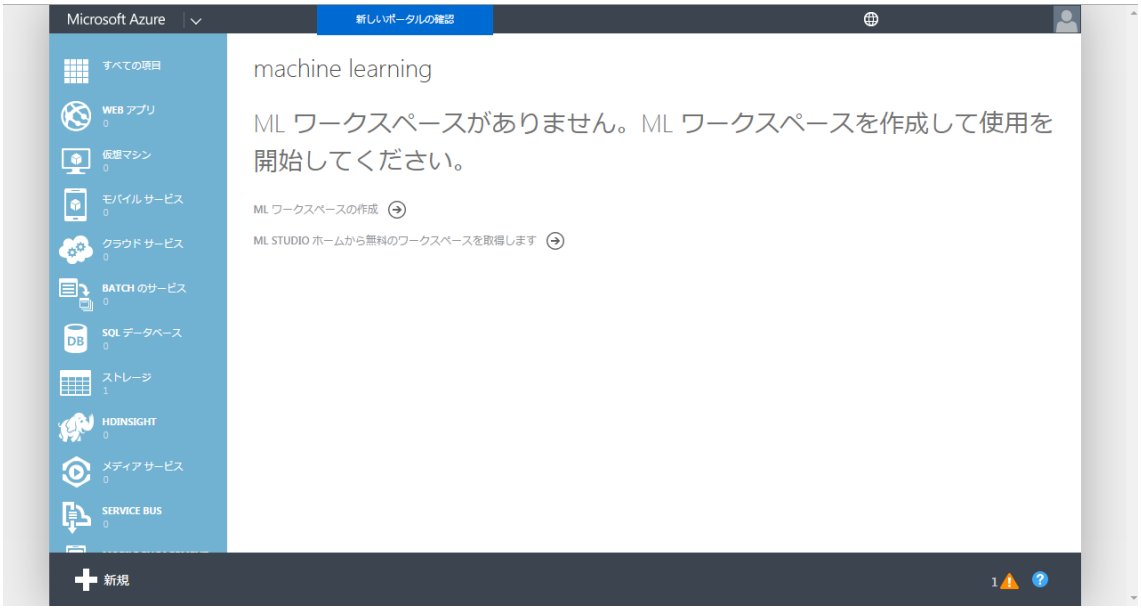

図 4: Azure ML ホーム画面

[ML ワークスペースの作成]をクリックし、新規 ML ワークスペース作成画面で任意のワークスペース名、ストレージアカウント名を入力します。場所は"Southeast Asia"がいいでしょう。[ML ワークスペースの作成]をクリックするとワークスペースの作成が開始されます。ワークスペースの作成が完了したら[STUDIO で開く]をクリックしてワークスペースを表示します(図 5)。

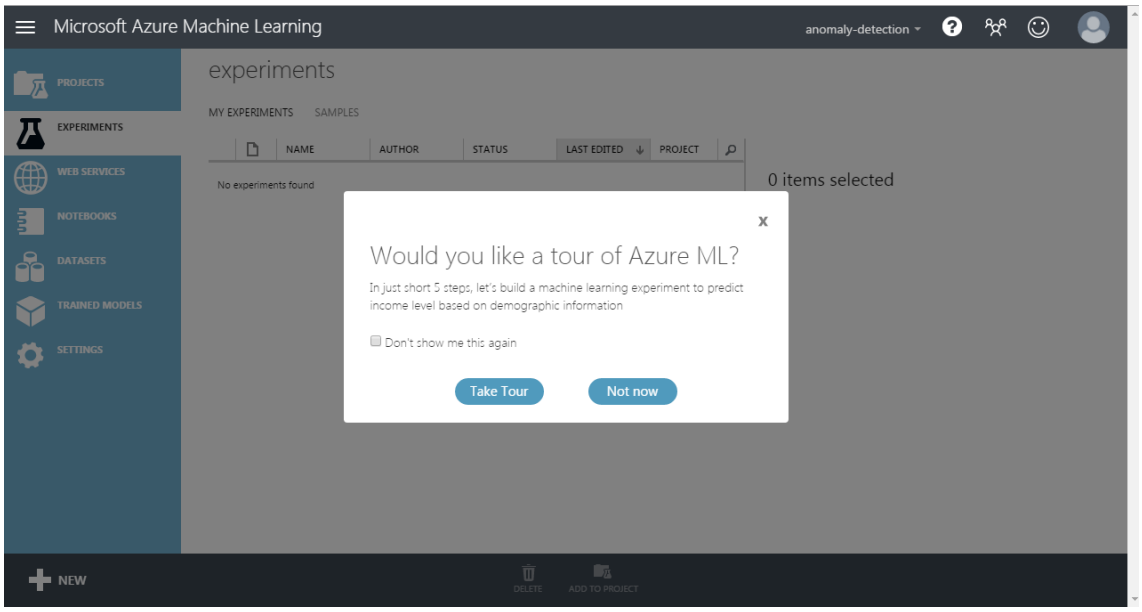

図 5: Azure ML ワークスペース画面

ポップアップを閉じると Experiment の作成画面が表示されます。まずは学習データとなる`trainingdata.csv`をアップロードします。[DATASET]を選択し[FROM LOCAL FILE]をクリックします。ファイル選択にて`trainingdata.csv`を選択して[OK(チェック)]をクリックします。

次に Experiment を作成します。[NEW]をクリックしたあとに[EXPERIMENT]を選択し、[Blank Experiment]をクリックすると Experiment の実装画面が表示されます(図 6)。
この画面を簡単に説明すると、左の領域にはモジュールと呼ばれるデータセット、データ加工処理、機械学習アルゴリズム等の一覧があります。このモジュールを中央の領域にドラッグ&ドロップで配置し、連結することでデータの取得からデータの加工、機械学習アルゴリズムの適用、結果の生成に至る一連のフローを作成することができます。右の領域では選択しているモジュールの詳細情報が表示されます。下部にあるボタン群は Experiment の保存、実行、Web API 公開等が実施できます。

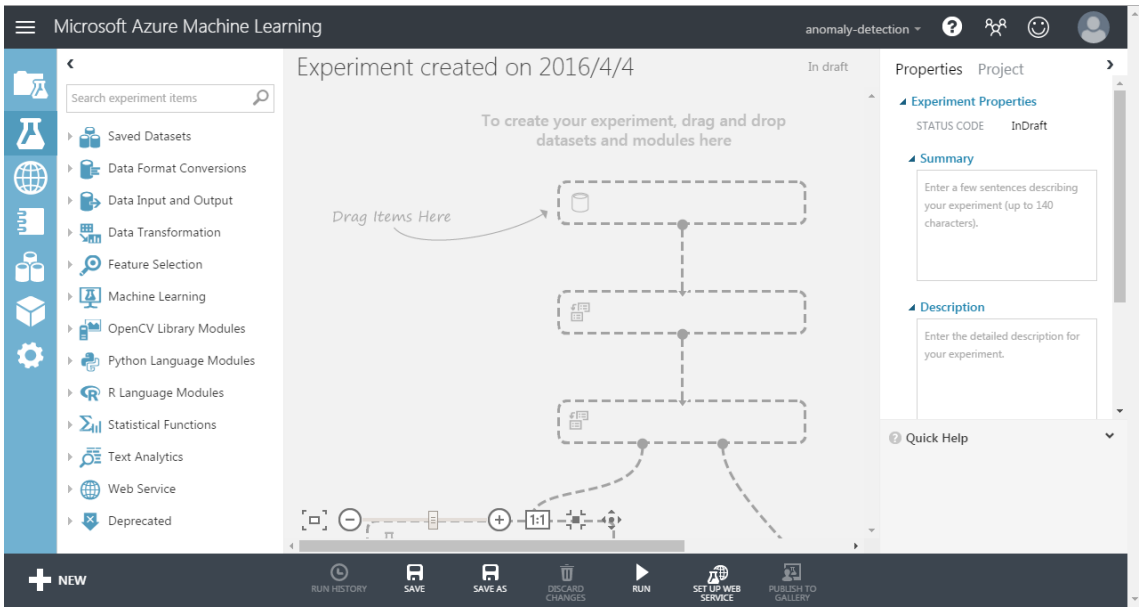

図 6: Azure ML Experiment 画面

先ほどアップロードした学習データと Azure ML が提供する異常検知アルゴリズムを用いてモデルを作成していきます。
まずは、Experiment 名を任意の名称に変更します。

モジュール一覧から[Saved Datasets] -> [My Datasets]から[trainingdata.csv]をドラッグ&ドロップで中央の領域に配置します。モジュールの下側にある○の部分(出力ポート)をクリックし[Visualize]を選択するとデータセットのデータを確認することができます(図 7)。これはデータセット以外のモジュールでも可能で、モジュールの処理結果を確認したい場合に便利です。

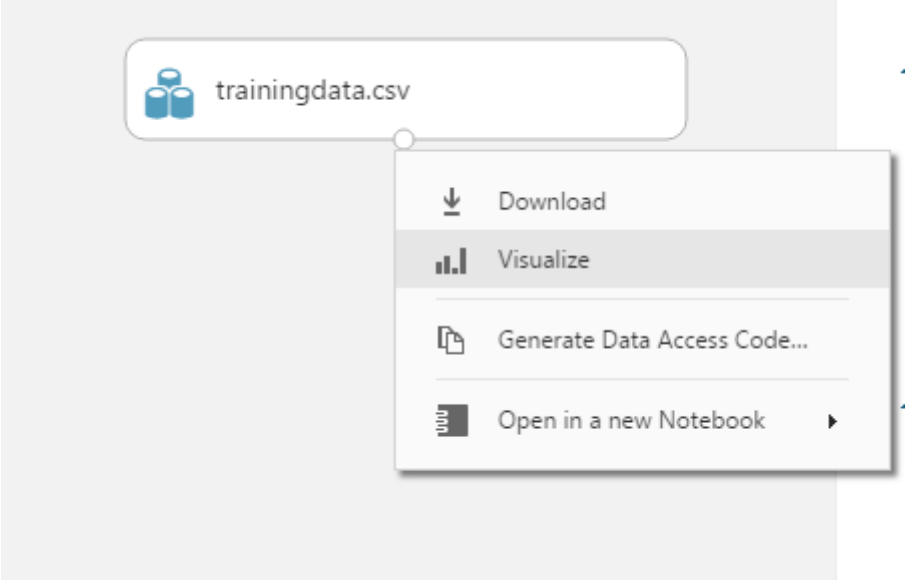

図 7: データセットの Visualize 選択画面

次に[Search experiment items]に"anomaly"と入力し、モジュールの中から[PCA-Based Anomaly Detection]と[Train Anomaly Detection Model]を図 8 のような構成になるように配置します。モジュールの連結は、モジュールの出力ポートをドラッグして連結したいモジュールにドロップします。[PCA-Based Anomaly Detection]を選択し、Properties の Number of components to use in PCA の値を 1 に変更します。(前述の PCA-Based Anomaly Detection の説明の例で言うと、テンプレートを一つしか使わないという意味です。もちろん、テンプレートをいくつ使いたいかは読者の自由です。)

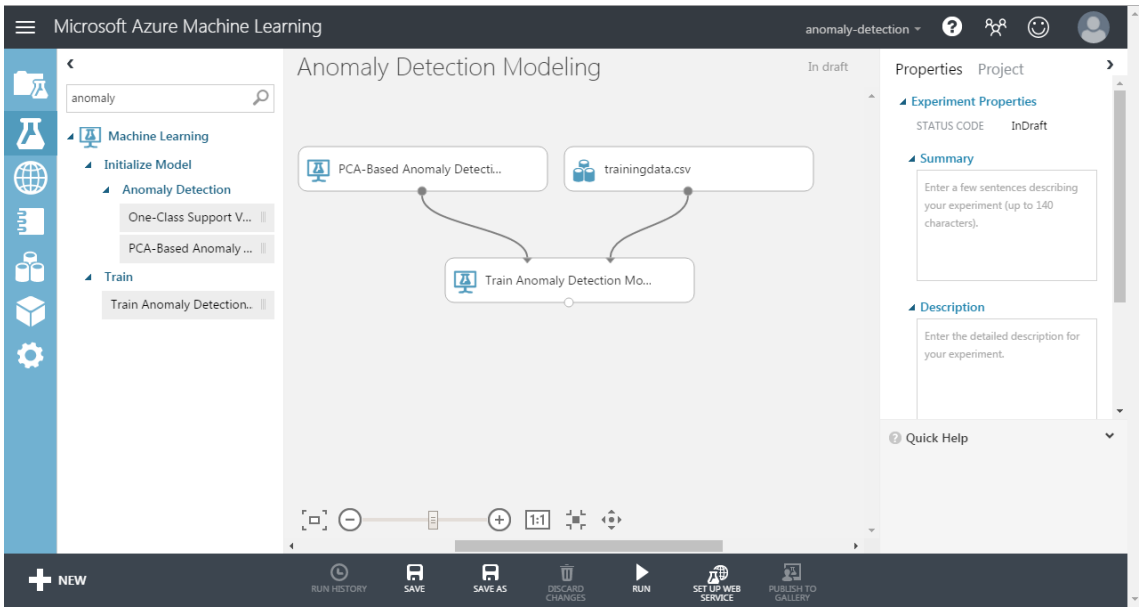

図 8: モデリング Experiment 画面

Experiment の実装が完了したら[RUN]をクリックします。ここで定義した各モジュールが実行され、学習データから学習モデルが作成されます。[Train Anomaly Detection Model]の出力ポートをクリックし[Save as Trained Model]を保存しましょう。これで次で作成する異音検知の Experiment からこの学習モデルを使用することができます。

ステップ 3　判定処理用訓練済みモデルの作成と公開

●SQL データベース作成

（まず、判定結果を蓄積し、ウェブアプリから非同期に利用するためのデータベースを構築します。データベースには Azure の SQL データベースを利用します。）

Azure ポータルのメニューから[SQL データベース]をクリックし、次の画面で[追加]をクリックします。任意の名前とリソースグループを入力し、価格レベルは"S1 Standard"に変更します。サーバーは新しく作成し、任意のサーバー名、サーバー管理者ログイン、パスワードを入力します(図 9)。場所は東日本を選択しましょう。入力が完了したら[作成]をクリックしてデータベースを作成します。

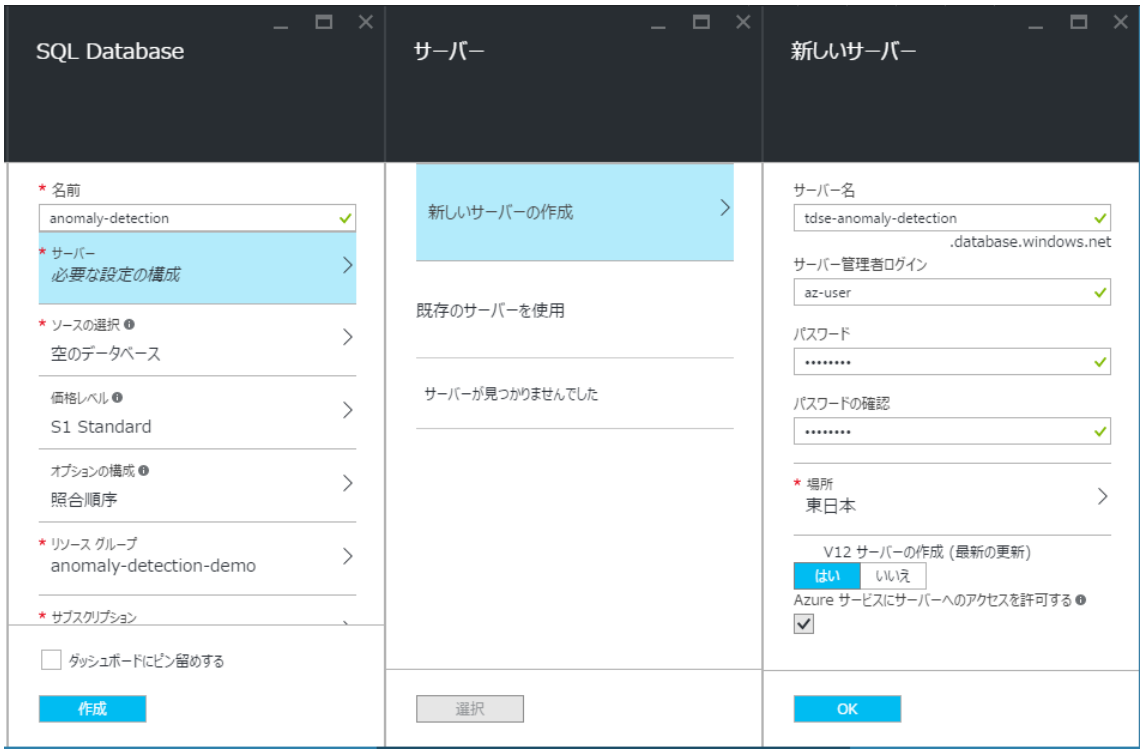

図 9: SQL データベース入力画面

次にテーブルを作成します。SQL データベースはデフォルトの設定でローカルマシンからアクセスできないため、ファイアウォールの設定を追加します。SQL データベースのメニューから作成したリソースを選択し、次の画面でサーバー名をクリックすると SQL Server の画面が表示されます。設定一覧にあるファイアウォールをクリックするとファイアウォール設定画面が表示されます。[クライアント IP の追加]をクリックし、[保存]をクリックします。これでローカルマシンからアクセスできるようになりました。

SQL(テーブル作成とデータ参照)を実行できるクライアントツールを用意します。SQL Server Management Studio や Visual Studio 等の IDE に付属している DB クライアントツールで構いません。
ファイル一式 ZIP の`azure-etc/database/ddl/create-tables.sql`を実行して`spectrum`テーブルと`anomaly`テーブルを作成します。`spectrum`テーブルはスペクトルを登録し、`anomaly`テーブルはスペクトルの判定結果を登録します。

●判定用訓練済みモデル Experiment 作成

（異音検知の Experiment は、判定対象となるスペクトルとその録音日時を入力値として処理するようにします。ファイル一式 ZIP の`azure-etc/machine-learning`ディレクトリにある`sample_spectrum.csv`と`sample_time.csv`がそれぞれスペクトルと録音日時に該当します。）

ワークスペースにアクセスし、先ほどと同様の手順で 2 つのデータセットをアップロードします。Experiment の作成も同様の手順を実施し、任意の Experiment 名を入力します。

次に図 10 のようにモジュールを配置してスコアリング処理まで定義します。[Trained model(saved from ...)]は上記で作成した学習モデルです。[Score Model]で学習モデルと`sample_spectrum.csv`で異常検知のスコアリングを行います。[Project Columns]はカラムを抽出することができます。左は`anomaly`テーブルを登録するためにスコアリング結果を持つカラムを抽出しており、右は`spectrum`テーブルを登録するためにスコアリング結果を持つカラムを除外しています。[Project Columns]の Properties で[Launch column selector]をクリックし、図 11,12 のように 2 つのモジュールを設定します。

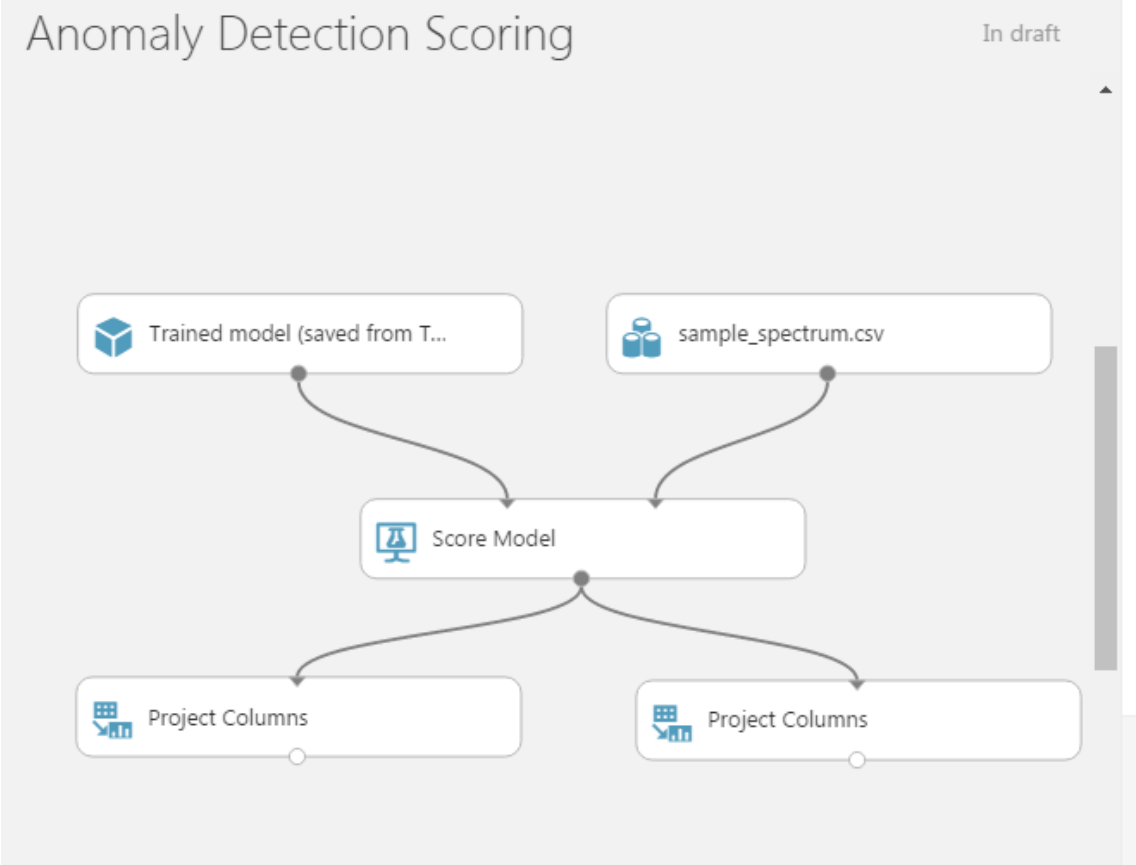

図 10: 異常判定 Experiment (スコアリング)

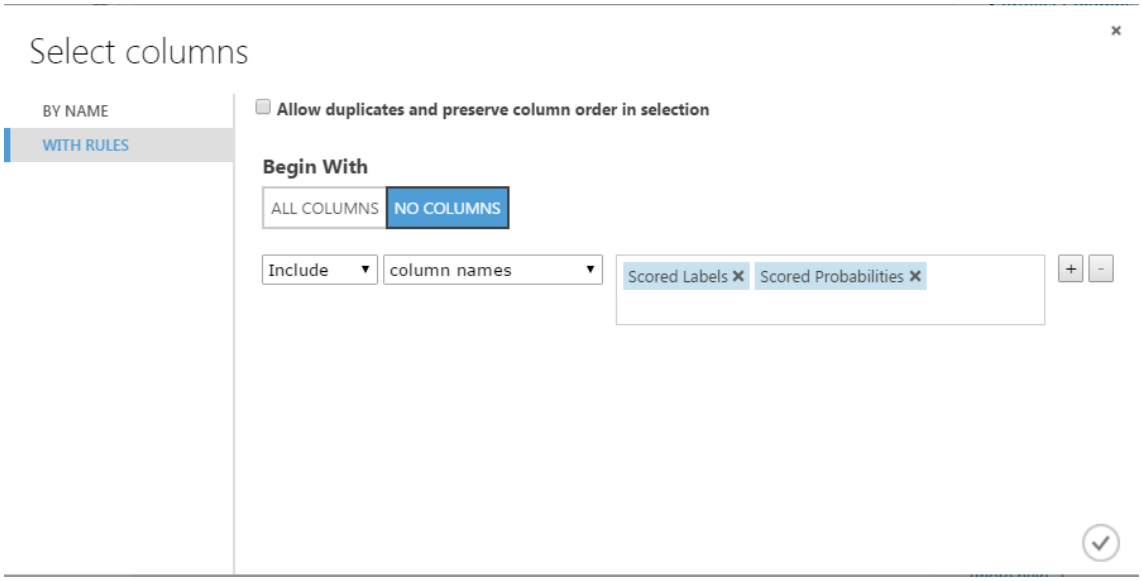

図 11: 異常判定 Experiment (スコアリング) Project Columns 設定(左)

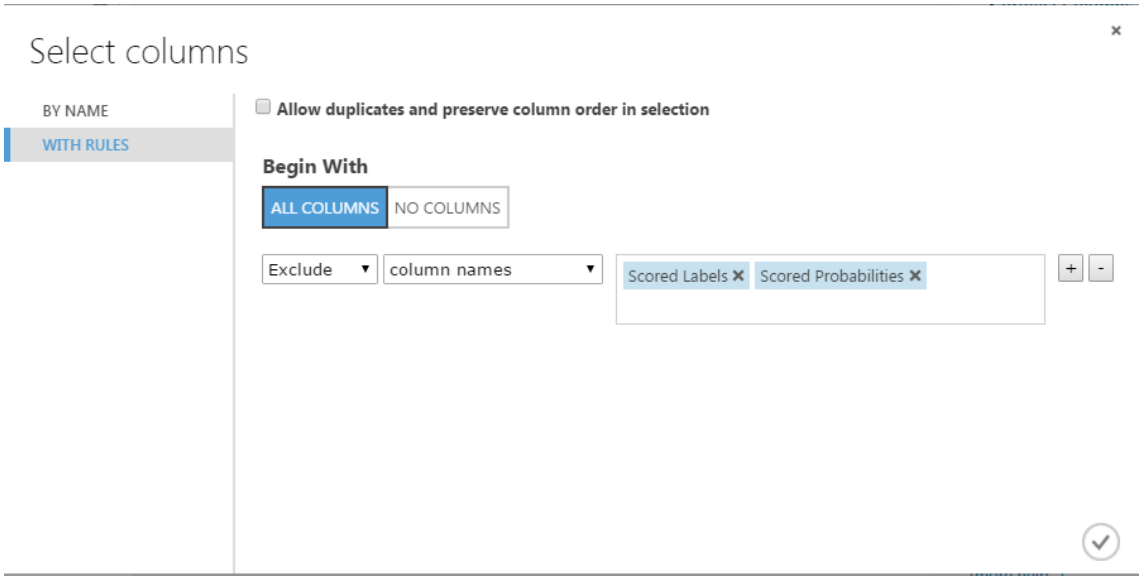

図 12: 異常判定 Experiment (スコアリング) Project Columns 設定(右)

ここで[RUN]をクリックし[Project Columns]の出力ポートで Visualize で結果を確認します。

続いて図 13 のようにモジュールを配置して結果をデータベースに登録するところまで定義します。[Execute R Script]はデータベースに登録するためにデータを整形しています。[Writer]で SQL データベースに登録しています。

左の[Execute R Script]の Properties にファイル一式 ZIP の`azure-etc/machine-learning/left_side.R`の内容を入力します。ここでは[Score Model]で出力された"Scored Labels"の値は使用せずに"Scored Probabilities"の値が 0.5 以上の場合に異常と判定しています。右の[Execute R Script]の Properties に`azure-etc/machine-learning/right_side.R`の内容を入力します。
左の[Writer]の Properties にデータベースへの接続情報とマッピング情報を入力します。Please specify data destination に"Azure SQL Database"を選択し、サーバー名、データベース名、ユーザ名、パスワードは上記で作成したデータベースの情報を入力します。

残りの項目はデータベースのテーブル、カラムへのマッピング情報を次のように入力します。
* Comma separated list of columns to be saved: time,Scored Probabilities,Scored Labels
* Data table name: anomaly
* Comma separated list of datatable columns: recorded_at,anomaly_score,is_anomaly

右の[Writer]も同様に接続情報を入力し、マッピング情報を次のように入力します。

* Comma separated list of columns to be saved: time,nrow,spectrum
* Data table name: spectrum
* Comma separated list of datatable columns: recorded_at,x,y
* Number of rows written per SQL Azure operation: 200

[sample_time.csv]の次にある[Project Columns]は、単純に実行するだけであれば不要ですが Web API 化するときに必要になります。"time"カラムを抽出するように設定しておきます。

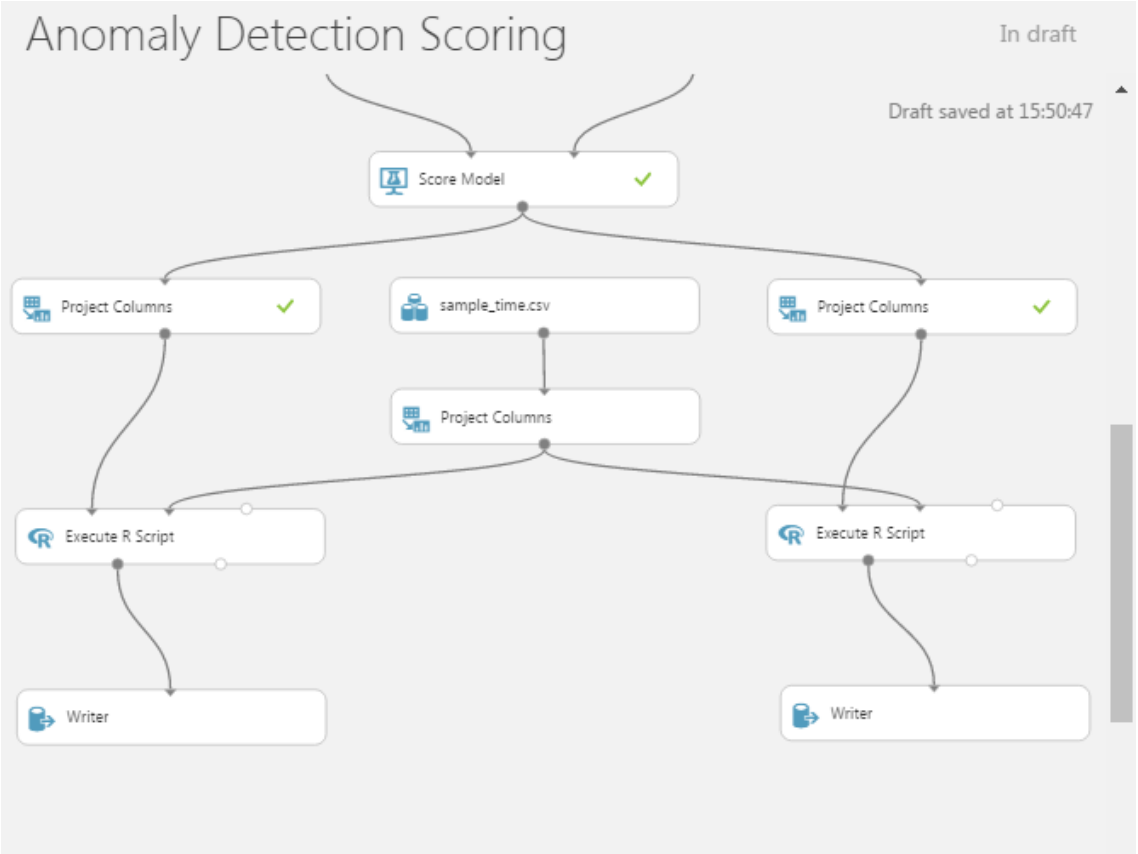

図 13: 異常判定 Experiment (データベース)

これで異音検知の Experiment が完成です。[RUN]をクリックして実行し`anomaly`テーブルに 1 件、`spectrum`テーブルに 200 件のレコードが登録されていることを確認します。

●Web API 公開

（ラズベリー・パイから実行できるように Web API として公開します。）
[SET UP WEB SERVICE]をクリックすると自動的に Web API の入出力モジュールである[Web service input]と[Web service output]が配置されます。入力はスペクトルと日時、出力は判定結果を定義したいので図 14 のように変更します。2 つの入力モジュールの内、[Score Model]に連結する入力モジュールの名称が"input1"、[Project Columns]に連携する入力モジュールの名称が"input2"となるように配置してください。入力モジュールは 2 つのモジュールに連結できないので日時入力モジュール(input2)は[Project Columns]に連結し、[Project Columns]から 2 つの[Execute R Script]に連結させています。

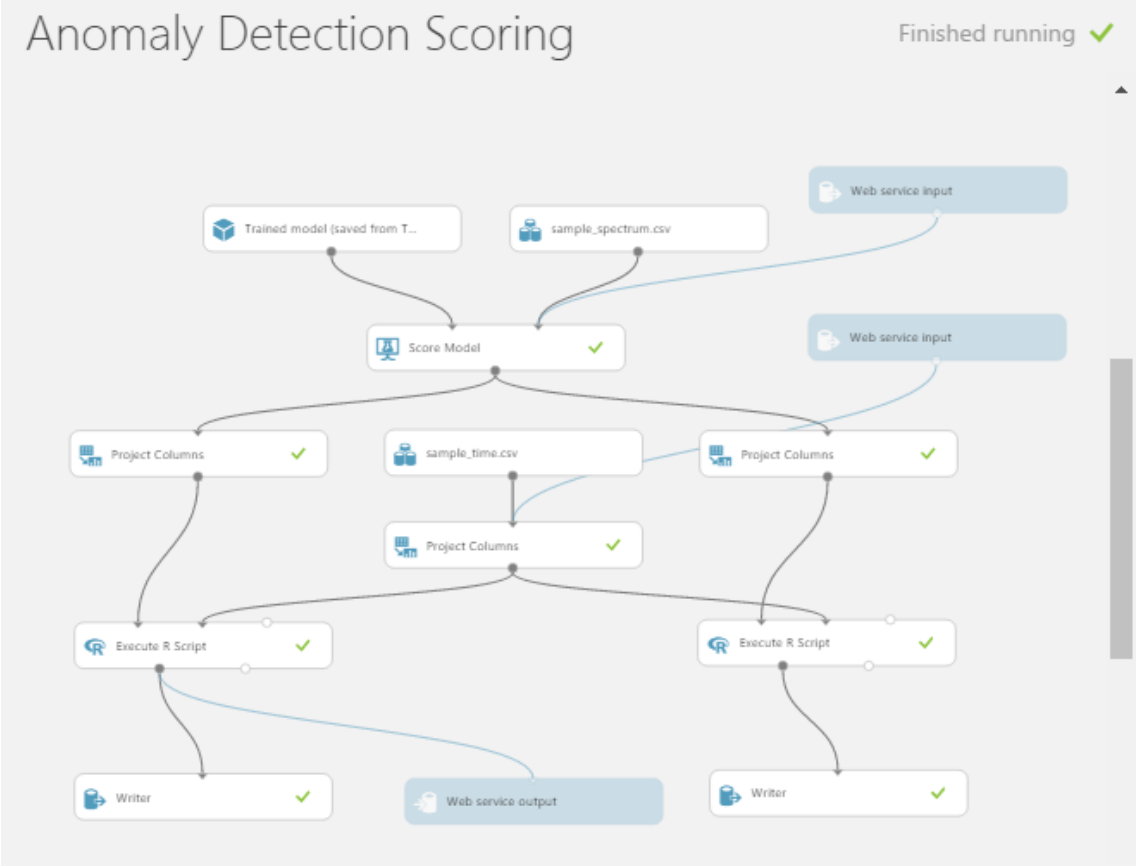

図 14: 異常判定 Experiment (Web API)

入出力の定義が完了したら[DEPLOY WEB SERVICE]をクリックします。Web API が公開され、Web サービスの画面が表示されます。API Key は Web API を呼び出すときに必要になるので保持しておきます。[REQUEST/RESPONSE]リンクをクリックすると Web API の詳細画面が表示されます。Request URI の内容が Web API を呼び出すときのエンドポイントになるので保持しておきます。

ステップ 5　異音検知の結果をパソコンから見られるようにする

●Web アプリの構築手順

Azure ポータルのメニューから[App Service]をクリックし、次に表示される画面で[追加]をクリックします。任意の名前とリソースグループを入力します。App Service プラン/場所は新規作成します。任意の App Service プラン名を入力し、場所は"Japan East"を選択します。価格レベルは、[すべて表示]をクリックし、"F1 Free"を選択します。入力が完了したら[作成]をクリックします。

次に Web Apps の設定を行います。作成した Web Apps を選択すると詳細画面が表示されます。設定の中から[アプリケーション設定]を選択します。この Web アプリは Java で実装されたものを利用するため、Java バージョンに"Java 8"を選択します。続いて下の方にスクロールするとアプリ設定というキーと値を入力できる項目があります。ここに上記で作成したデータベースへの接続情報を設定します(図 15)。

* DB_HOSTNAME: [DB ホスト名]
* DB_NAME: [DB 名]
* DB_USER: [DB ユーザ名]
* DB_PASSWORD: [DB パスワード]

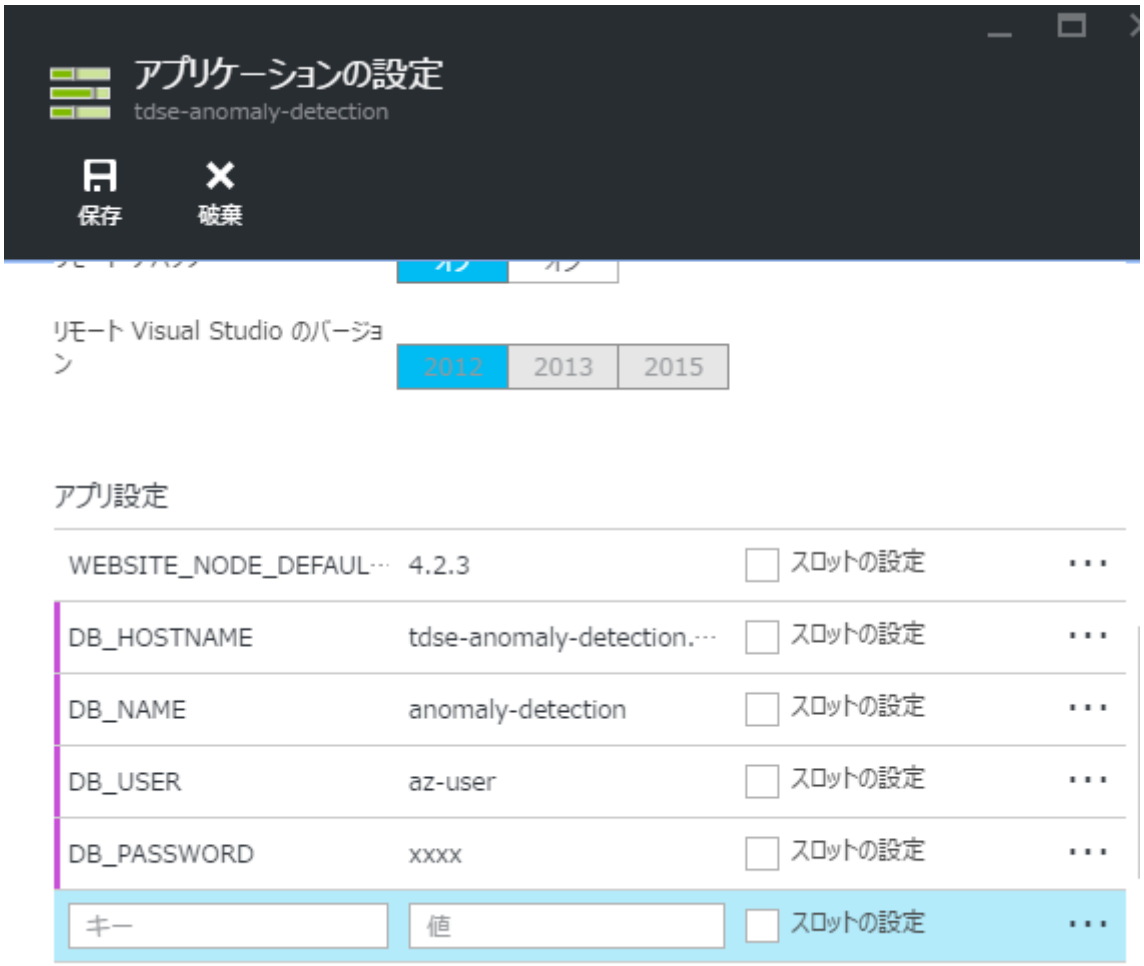

図 15: Web Apps アプリ設定

その他の項目はデフォルト値で問題ないので[保存]をクリックします。

●Web アプリのビルド、デプロイ

ファイル一式 ZIP の`anomaly-monitoring`ディレクトリが Web アプリのソースコードです。Java で実装されているため、ビルドに必要な Java Development Kit と Maven の最新版をローカルマシンにインストールします。

次にデータベースに接続する JDBC ドライバーを以下の URL からダウンロードします。バージョンがいくつかありますが最新のもので構いません。

https://www.microsoft.com/ja-JP/download/details.aspx?id=11774

インストールまたは、解凍すると生成される"sqljdbcXX.jar"(XX はバージョン)が JDBC ドライバーです。このファイルを"sqljdbc.jar"にリネームして`lib`ディレクトリに配置します。

`pom.xml`ファイルがあるディレクトリに移動して次のコマンドを実行します。初回は Web アプリの依存ライブラリやビルドに必要なライブラリをダウンロードするため少し時間がかかるかもしれません。

```
$ mvn clean package
(略)
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 6.891 s
[INFO] Finished at: 2016-04-05T11:24:10+09:00
[INFO] Final Memory: 31M/212M
[INFO] ------------------------------------------------------------------------
```

ビルドに成功すると上記のように"BUILD SUCCESS"と表示されます。`target`ディレクトリはビルドによって生成された実行ファイルや中間ファイル等が格納されています。`target`ディレクトリにある`statusbelt-azure.zip`を Azure Web Apps にデプロイしていきます。

Web Apps へのデプロイ方法はいくつかありますが、今回はブラウザからファイルをアップロードする方式にします。以下の URL の[webappname]の部分を Web Apps の名称に置き換えてアクセスすると図 16 のような画面が表示されます。
https://[webappname].scm.azurewebsites.net/

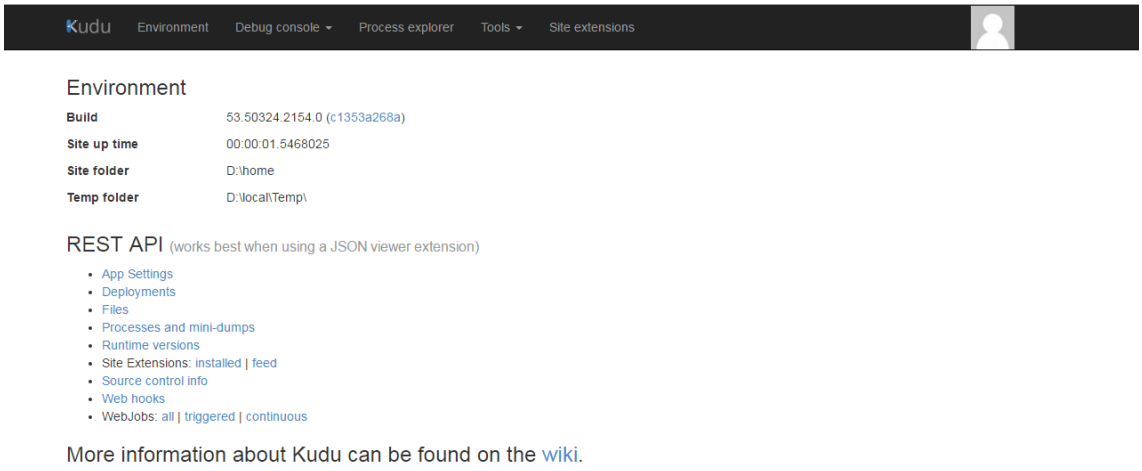

図 16: Web Apps Kudu 画面

メニューの[Debug console] -> [CMD]をクリックすると、現在の階層のフォルダ一覧が表示されます。[site] -> [wwwroot]をクリックすると、D:¥home¥site¥wwwroot の階層に辿り着いたと思います。上記でビルドした`statusbelt-azure.zip`を図 17 のようにドラッグ&ドロップすると zip 内のファイルが展開されます。次回以降は WebApps を停止してからデプロイしてください。

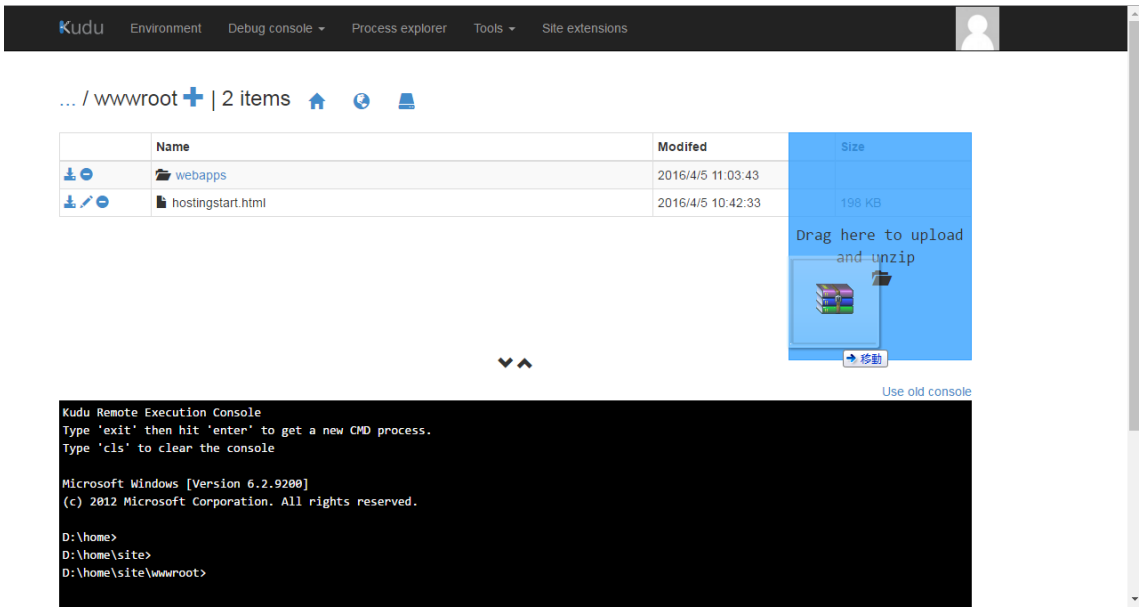

図 17: Web Apps デプロイ