Modern AI: Challenges and Principles

Monday, August 3, 2020 1:59 PM

- Responsible AI
  - Why is responsibility a concern?
    - Poorly curated ML models can increase or induce inequality in demographics
      - Ex. Healthcare datasets with features that were proxies for wealth bias against the poor
    - AI being weaponized to improve vectors of attack in cybersecurity
      - Ex. AI being used to probe vulnerable groups to different types of scams (email etc)
    - AI can introduce unintentional bias
      - Ex. Online advertising system showing ads for high income jobs to men more than women
    - Adversarial Attacks
      - Ex. Autonomous vehicle being tricked to see a stop sign as something else
    - Killer Drones
      - Moral question of delegating killing and harming humans to a ML model
    - Deep Fakes
      - Fake videos, weaponizes misinformation, threatens outlets of media and truth
    - Data poisoning
      - Intentionally manipulating data and public datasets to skew ML models one way or another
    - Hype
      - Lots of hype drives unrealistic expectations. Dangers of minimizing ML limitations and maximizing promises.
  - Approaches to Responsible AI
    - Model Explainability
      - Be able to interpret and explain behavior of trained model
      - Generates insights to validate model behavior and hypotheses, build trust with beneficiaries of model
      - Two key aspects
        - Global Behavior: explanation of how model works in general
        - Local Explanations: explanation of how a certain prediction or set of predictions is reached
          - Why it makes certain predictions for specific inputs
    - Fairness
      - Who might this harm? Who is neglected? Is there any bias in who is favored? Is anyone misrepresented?

- Microsoft AI Principles
  - Six Principles for Foundation and Development of Ai-Powered Solutions
    - Fairness
      - All systems should treat all people fairly and not affect similarly situated groups in different ways
        - Use data pools and analytical techniques that eliminate bias
        - Requires systematic review and curation of data and models
    - Reliability and Safety
      - Customers need to trust an AI solution will perform reliably and safely within a certain set of parameters
      - Requires extensive testing of training dat and models, robust feedback system and ways of documenting and auditing performance
      - Determine when AI system should seek input of humans
    - Privacy and Security
      - AI systems should be secure and respect existing privacy laws
      - AI systems should be transparent about data collection and have good controls
    - Inclusiveness
      - AI should include and engage people
      - Use inclusive design practices to eliminate unintentional barriers
      - Understand context, needs and expectations of users and address barriers that could potentially exclude people
    - Transparency
      - When AI helps make decisions that impact people, people should understand how those decisions are made
      - How models interact with data and make decisions should be visible and understandable to users and the people it affects
    - Accountability
      - Designers and deployers of Ai systems should be held accountable for the effects and impact of their models and how they operate
      - Periodic checks for accountability adherence
    - Transparency and Accountability are foundational principles that affect the others
  - PARFIT - 'perfect' in old French -> "A part of perfection is addressing 'parfit' principles of AI"
    - Privacy and security
    - Accountability
    - Reliability and safety
    - Fairness
    - Inclusiveness
    - Transparency

Model Transparency and Explainability

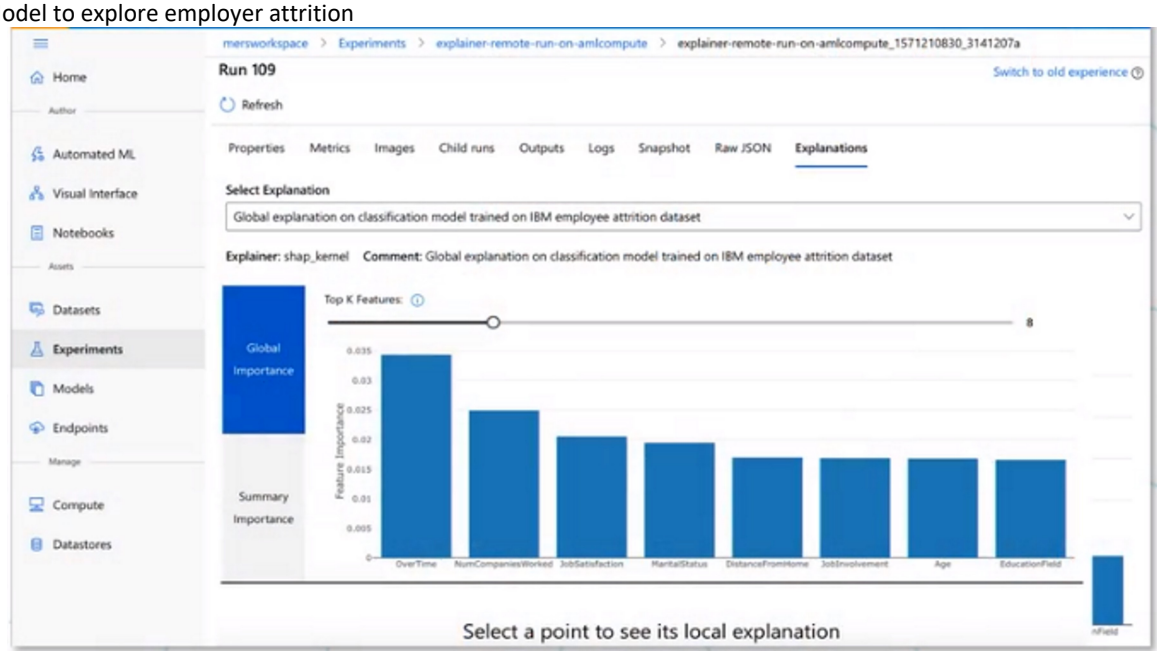
Monday, August 3, 2020 4:25 PM

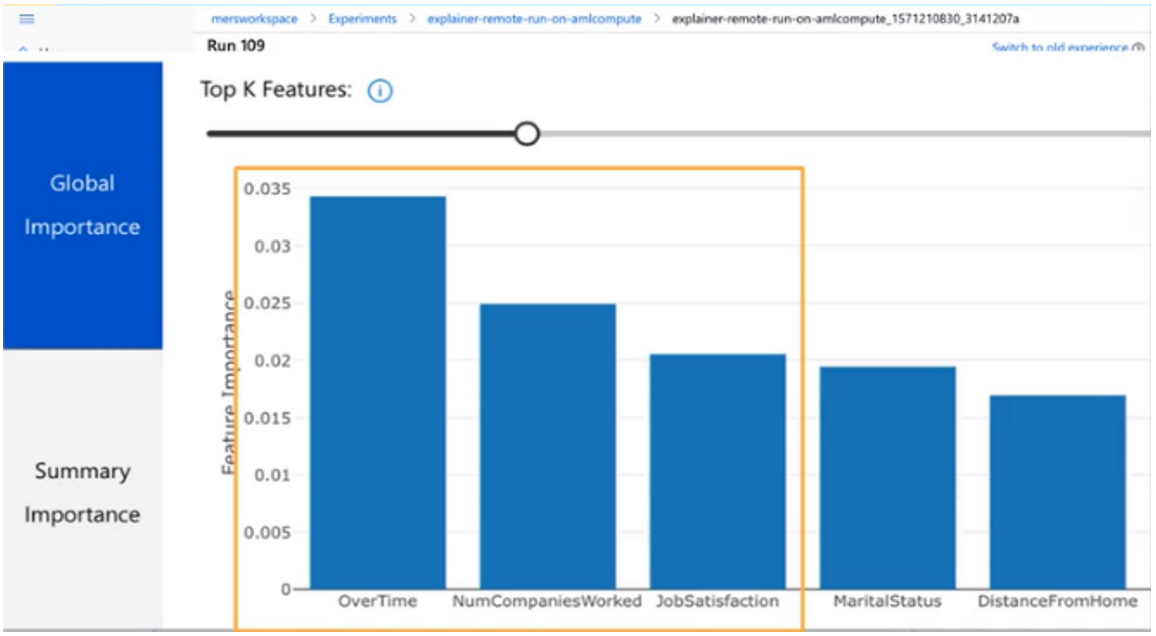
- Model Transparency
  - Model training has a certain degree of opacity in how it takes in data and then returns predictions
  - Opacity is the degree to which the inenr workings of a function can be seen, explained and understood
    - Depends on class of algorithm used to train models
    - Spectrum of opacity and ease of explanation
      - 

- Model Explainability
  - Feature importance is a key component
    - Explaining how certain features factor in and how important each one is to the resulting predictions
    - Ex. Determining which features matter more in predicting bike rentals
      - 

  - Models can be explained using 'explainers'
    - Two kinds of explainers
      - Direct Explainers
        - Use direct explainer when you know specific tool best for the job
        - Choose direct explainer based on model type then use in code to explain model
        - Model specific direct explainers are tied to their corresponding model
        - Model agnostic direct explainers can be used to explain any model regardless of algorithm used to train it
          - Approximation these create are readily explainable and can be used for many models
Examples of model specific direct explainers	Examples of model agnostic direct explainers
SHAP Tree Explainer	Mimic Explainer
SHAP Deep Explainer	SHAP Kernel Explainer
      - Meta Explainers
        - Used for automatic selection of direct explanations
        - Will select best direct explainer based on model and dataset to generate model information

- Meta explainers are like 'magic toolboxes' that give you the right tool for the explainer job (direct explainer is the tool)
- Large categories of meta explainers
Examples of Meta Explainers
Tabular Explainer
Text Explainer
Image Explainer

- Explainers are used to create global/local predictions in notebook code
  - Visualizations are then used to explore explanations graphically
  - Ex. Model to explore employer attrition
    - 

    - 

Lab: Model Explainability

Monday, August 3, 2020 4:36 PM

git clone https://github.com/solliancenet/udacity-intro-to-ml-labs.git LAB 23

Explaining models

Model interpretability with Azure Machine Learning service

Machine learning interpretability is important in two phases of machine learning development cycle:

- During training: Model designers and evaluators require interpretability tools to explain the output of a model to stakeholders to build trust. They also need insights into the model so that they can debug the model and make decisions on whether the behavior matches their objectives. Finally, they need to ensure that the model is not biased.
- During inferencing: Predictions need to be explainable to the people who use your model. For example, why did the model deny a mortgage loan, or predict that an investment portfolio carries a higher risk?

The Azure Machine Learning Interpretability Python SDK incorporates technologies developed by Microsoft and proven third-party libraries (for example, SHAP and LIME). The SDK creates a common API across the integrated libraries and integrates Azure Machine Learning services. Using this SDK, you can explain machine learning models globally on all data, or locally on a specific data point using the state-of-art technologies in an easy-to-use and scalable fashion.

Overview

In this lab, we will be using a subset of NYC Taxi & Limousine Commission - green taxi trip records available from Azure Open Datasets. The data is enriched with holiday and weather data. We will use data transformations and the GradientBoostingRegressor algorithm from the scikit-learn library to train a regression model to predict taxi fares in New York City based on input features such as, number of passengers, trip distance, datetime, holiday information and weather information.

The primary goal of this quickstart is to explain the predictions made by our trained model with the various Azure Model Interpretability packages of the Azure Machine Learning Python SDK.

Predicting NYC Taxi Fares And Model Explainer

In this quickstart, we will be using a subset of NYC Taxi & Limousine Commission - green taxi trip records available from Azure Open Datasets. The data is enriched with holiday and weather data. We will use data transformations and the GradientBoostingRegressor algorithm from the scikit-learn library to train a regression model to predict taxi fares in New York City based on input features such as, number of passengers, trip distance, datetime, holiday information and weather information.

The primary goal of this quickstart is to explain the predictions made by our trained model with the various Azure Model Interpretability packages of the Azure Machine Learning Python SDK.

Install required libraries

After your install these libraries it is recommended that you **restart** the notebook kernel from the **Kernel** menu above. After restarting the kernel, start from the **Azure Machine Learning and Model Interpretability SDK-specific Imports** section.

You can ignore any incompatibility errors. Please run the cell below only once.

In [ ]:
```
!pip install --upgrade interpret-community
!pip install flask-cors
```

Azure Machine Learning and Model Interpretability SDK-specific Imports

Remember to restart the kernel before proceeding.

Run the following cell to import the modules used in this notebook.

In [3]:
```
import os
import numpy as np
import pandas as pd
import pickle
import sklearn
from sklearn.externals import joblib
import math

print("pandas version: {} numpy version: {}".format(pd.__version__, np.__version__))

sklearn_version = sklearn.__version__
print('The scikit-learn version is {}.'.format(sklearn_version))

import azureml
from azureml.core import Workspace, Experiment, Run
from azureml.core.model import Model

from interpret.ext.blackbox import TabularExplainer
from azureml.interpret.scoring.scoring_explainer import TreeScoringExplainer, save

print('The azureml.core version is {}.'.format(azureml.core.VERSION))
```

```
pandas version: 0.23.4 numpy version: 1.16.2
The scikit-learn version is 0.20.3.
The azureml.core version is 1.9.0.
```

Setup

To begin, you will need to provide the following information about your Azure Subscription.

In the following cell, be sure to set the values for `subscription_id`, `resource_group`, `workspace_name` and `workspace_region` as directed by the comments (*these values can be acquired from the Azure Portal*).

You can get all of these values from the lab guide on the right:

1. In the tabs at the top of the lab guide, select `Environment Details`.
2. Copy the values from SubscriptionID, ResourceGroup, WorkspaceName and WorkspaceRegion and paste them as the values in the cell below.

Execute the following cell by selecting the `>|Run` button in the command bar above.

In [4]:
```
#Provide the Subscription ID of your existing Azure subscription
subscription_id = "55e71b9d-a209-42c0-8818-ca9cc885909c" # <- needs to be the subscription within the Azure resource group for th

#Provide values for the existing Resource Group
resource_group = "aml-quickstarts-62958" # <- enter the name of your Azure Resource Group

#Provide the Workspace Name and Azure Region of the Azure Machine Learning Workspace
workspace_name = "quick-starts-ws-62958" # <- enter the name of the Azure Machine Learning workspace
workspace_region = "SouthCentralUS" # <- region of your Azure Machine Learning workspace

experiment_name = "lab-explainability"
```

Create and connect to an Azure Machine Learning Workspace

Run the following cell to connect to your existing Azure Machine Learning **Workspace** and save the configuration to disk (next to the Jupyter notebook).

Important Note: You may be prompted to login in the text that is output below the cell. If you are, be sure to navigate to the URL displayed and enter the code that is provided. Once you have entered the code, return to this notebook and wait for the output to read `Workspace configuration succeeded`.

In [6]:
```
ws = Workspace.create(
```

Visualizing the Global Explanation

Run the following cell to create a dashbaord the enables you to explore the data and visualize the explanation.

In the Dashboard that is displayed, try answering the following questions:

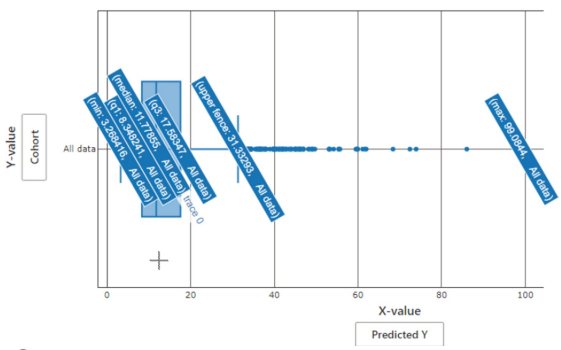
1. Select the `Data Exploration` tab, the set the `X value` to `tripDistance` and the `Y value` to `PredictedY` (this is predicted fare mount). What happens to the predicted fare as the trip distance increases?
2. Select the `Global Importance` tab. Drag the slider under Top K Features so its value is set to `3`. What are the top 3 most important features? Which feature has the highest feature importance (and is therefore the most important feature)?

In [10]:
```
from interpret_community.widget import ExplanationDashboard

ExplanationDashboard(global_explanation, model=clf, datasetX=X_test)
```

Open in new tab

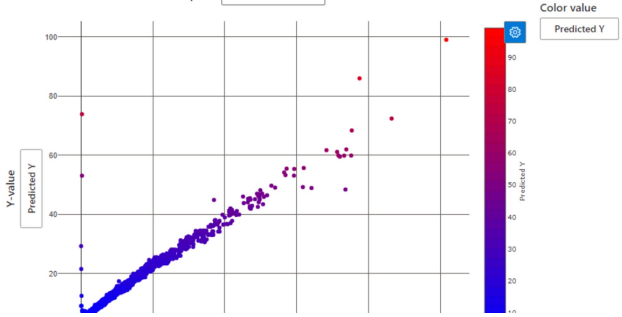
Evaluate the performance of your model by exploring the distribution of your prediction values and the values of your model performance metrics. You can further investigate your model by looking at a comparative analysis of its performance across different cohorts or subgroups of your dataset. Select filters along y-value and x-value to cut across different dimensions. Select the gear in the graph to change graph type.



Model performance statistics require the true outcomes be provided in addition to the predicted outcomes

Explore your dataset statistics by selecting different filters along the X, Y, and color axis to slice your data along different dimensions. Create dataset cohorts above to analyze dataset statistics with filters such as predicted outcome, dataset features and error groups. Use the gear icon in the upper right-hand corner of the graph to change graph types.

Select a dataset cohort to explore: All data



Run the following cell to connect to your existing Azure Machine Learning **Workspace** and save the configuration to disk (next to the Jupyter notebook).

Important Note: You may be prompted to login in the text that is output below the cell. If you are, be sure to navigate to the URL displayed and enter the code that is provided. Once you have entered the code, return to this notebook and wait for the output to read `Workspace configuration succeeded`.

In [6]:
```
ws = Workspace.create(
    name = workspace_name,
    subscription_id = subscription_id,
    resource_group = resource_group,
    location = workspace_region,
    exist_ok = True)

ws.write_config()
print('Workspace configuration succeeded')
```

```
Workspace configuration succeeded
```

Train the Model

Run the following cell to download the dataset, split the data into training and test sets and create a pipeline that includes a few steps to clean and standardize the data and ultimately train the model.

NOTE: Do not get too concerned about the details of the following code. If you take anything away from this cell, it should be that a model has been trained and stored in the variable `clf`.

In [7]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn_pandas import DataFrameMapper
from sklearn.metrics import mean_squared_error

data_url = ('https://introtomlsampledata.blob.core.windows.net/data/nyc-taxi/nyc-taxi-sample-data.csv')

df = pd.read_csv(data_url)
x_df = df.drop(['totalAmount'], axis=1)
y_df = df['totalAmount']

X_train, X_test, y_train, y_test = train_test_split(x_df, y_df, test_size=0.2, random_state=0)

categorical = ['normalizeHolidayName', 'isPaidTimeOff']
numerical = ['vendorID', 'passengerCount', 'tripDistance', 'hour_of_day', 'day_of_week',
             'day_of_month', 'month_num', 'snowDepth', 'precipTime', 'precipDepth', 'temperature']

numeric_transformations = [([f], Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())])) for f in numerical]

categorical_transformations = [([f], OneHotEncoder(handle_unknown='ignore', sparse=False)) for f in categorical]

transformations = numeric_transformations + categorical_transformations

clf = Pipeline(steps=[('preprocessor', DataFrameMapper(transformations)),
                      ('regressor', GradientBoostingRegressor())])

clf.fit(X_train, y_train)

y_predict = clf.predict(X_test)
y_actual = y_test.values.flatten().tolist()
rmse = math.sqrt(mean_squared_error(y_actual, y_predict))
print('The RMSE score on test data for GradientBoostingRegressor: ', rmse)
```

```
The RMSE score on test data for GradientBoostingRegressor:  4.171593931282874
```

Global Explanation Using a Meta Explainer (TabularExplainer)

Global Model Explanation is a holistic understanding of how the model makes decisions. It provides you with insights on what features are most important and their relative strengths in making model predictions.

To initialize an explainer object, you need to pass your model and some training data to the explainer's constructor.

Note that you can pass in your feature transformation pipeline to the explainer to receive explanations in terms of the raw features before the transformation (rather than engineered features).

In [8]:
```
# "features" and "classes" fields are optional
trained_gradient_boosting_regressor = clf.steps[-1][1]
tabular_explainer = TabularExplainer(trained_gradient_boosting_regressor,
                                     initialization_examples=X_train,
                                     features=X_train.columns,
                                     transformations=transformations)
```

```
Setting feature_perturbation = "tree_path_dependent" because no background data was given.
```

TabularExplainer uses one of three explainers: TreeExplainer, DeepExplainer, or KernelExplainer, and is automatically selecting the most appropriate one for our use case.

You can learn more about the underlying model explainers at Azure Model Interpretability.

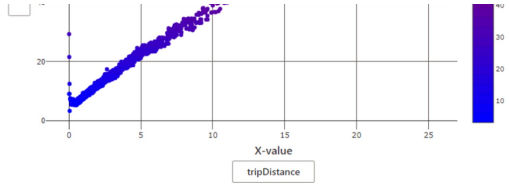
Get the global feature importance values

Run the below cell and observe the sorted global feature importance. You will note that `tripDistance` is the most important feature in predicting the taxi fares, followed by `hour_of_day`, and `day_of_week`.

In [9]:
```
# You can use the training data or the test data here
global_explanation = tabular_explainer.explain_global(X_test)

# Sorted feature importance values and feature names
sorted_global_importance_values = global_explanation.get_ranked_global_values()
sorted_global_importance_names = global_explanation.get_ranked_global_names()
dict(zip(sorted_global_importance_names, sorted_global_importance_values))
```

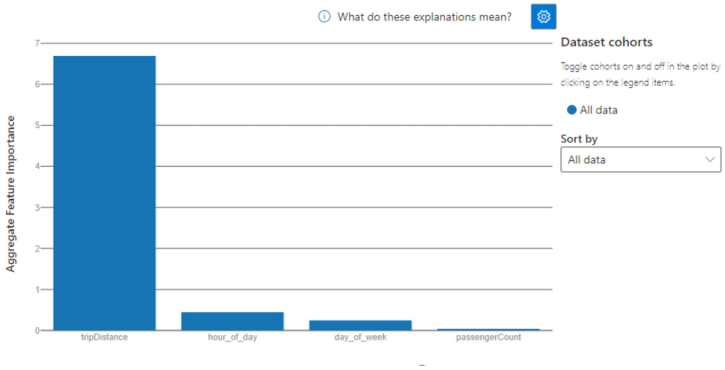
Out[9]:
```
{'tripDistance': 6.690378745302279,
 'hour_of_day': 0.44402071508583413,
 'day_of_week': 0.24710958587635976,
 'passengerCount': 0.0363110402551484,
 'precipDepth': 0.03502353421495021,
 'temperature': 0.028067339711470293,
 'day_of_month': 0.025269253140273364,
 'isPaidTimeOff': 0.016554471778010288,
 'snowDepth': 0.011707454642664875,
 'month_num': 0.006149237670690983,
 'vendorID': 0.0020700858868743423,
 'precipTime': 0.0009748171881798654,
 'normalizeHolidayName': 0.0}
```

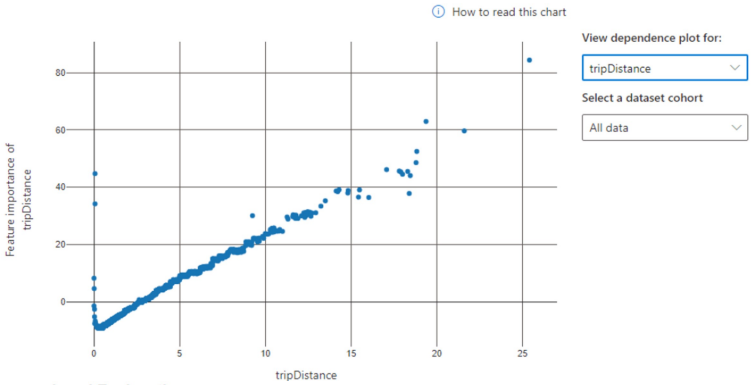


Model Performance | Dataset Explorer | **Aggregate Feature Importance** | Individual Feature Importance & What-If

Explore the top-k important features that impact your overall model predictions (a.k.a. global explanation). Use the slider to show descending feature importance values. Select up to three cohorts to see their feature importance values side by side. Click on any of the feature bars in the graph to see how values of the selected feature impact model prediction.

Top 1-4 features





Local Explanation

You can use the TabularExplainer for a single prediction. You can focus on a single instance and examine model prediction for this input, and explain why.

We will create two sample inputs to explain the individual predictions.

- **Data 1**
  - 4 Passengers at 3:00PM, Friday July 5th, temperature 80F, travelling 10 miles
- **Data 2**
  - 1 Passenger at 6:00AM, Monday January 20th, rainy, temperature 35F, travelling 5 miles

In [11]:
```
# Create the test dataset
columns = ['vendorID', 'passengerCount', 'tripDistance', 'hour_of_day', 'day_of_week', 'day_of_month',
           'month_num', 'normalizeHolidayName', 'isPaidTimeOff', 'snowDepth', 'precipTime',
           'precipDepth', 'temperature']

data = [[1, 4, 10, 15, 4, 5, 7, 'None', False, 0, 0.0, 0.0, 80],
        [1, 1, 5, 6, 0, 20, 1, 'Martin Luther King, Jr. Day', True, 0, 2.0, 3.0, 35]]

data_df = pd.DataFrame(data, columns = columns)
```

In [12]:
```
# explain the test data
local_explanation = tabular_explainer.explain_local(data_df)

# sorted feature importance values and feature names
sorted_local_importance_names = local_explanation.get_ranked_local_names()
sorted_local_importance_values = local_explanation.get_ranked_local_values()

# package the results in a DataFrame for easy viewing
results = pd.DataFrame([sorted_local_importance_names[0][0:5], sorted_local_importance_values[0][0:5],
                        sorted_local_importance_names[1][0:5], sorted_local_importance_values[1][0:5]],
                       columns = ['1st', '2nd', '3rd', '4th', '5th'],
                       index = ['Data 1', '', 'Data 2', ''])
print('Top 5 Local Feature Importance')
results
```

```
Top 5 Local Feature Importance
```

Out[12]:

	1st	2nd	3rd	4th	5th
Data 1	tripDistance	hour_of_day	passengerCount	day_of_week	temperature
	23.7359	0.817677	0.405282	0.131575	0.124938
Data 2	tripDistance	temperature	day_of_week	month_num	precipTime
	7.71963	0.0885763	0.077376	0.0178806	2.89782e-05

As we saw from the Global Explanation that the **tripDistance** is the most important global feature. Other than `tripDistance`, the rest of the top 5 important features were different for the two samples.

- Data 1: Passenger count 4 and 3:00 PM on Friday were also important features in the prediction.
- Data 2: The weather-related features (rainy, temperature 35F), day of the week (Monday) and month (January) were also important.

Model Fairness + Lab

Monday, August 3, 2020 4:52 PM

- Model Fairness
  - FairLearn Toolkit
    - Toolkit to identify and mitigate unfairness in machine learning models
    - https://github.com/fairlearn/fairlearn
      - Python package and Jupyter notebook with examples
    - Group Fairness: Which groups of individuals for experiencing harms?
      - Relevant groups need to be specified the Fairlearn probes model with data to identify groups
      - FairLearn provides support for assessing and mitigating fairness imbalances of your model
  - Demo of FairLearn
    - FairLearn GridSearch