# Taxi Fare Predictors

## Introduction to Big Data and Analytics
## CSCI 6444

**Team Members:**

| | |
|---|---|
| **Hrutik Naik** |  |
| **Shrumi Dedhia** |  |
| **Aditya Kumar** |  |

## Introduction:

The NYC TLC Taxi Fare Prediction project leverages an innovative ML pipeline that harnesses the power of Amazon SageMaker, AWS CodePipeline, and AWS CodeDeploy to predict taxi fares in New York City. With an emphasis on data-driven accuracy and efficiency, this project represents a crucial intersection of data science and real-world application. By incorporating a publicly available New York green taxi dataset, the project seeks to build a robust model that accurately estimates taxi fares, catering to the specific dynamics of the city's taxi services.

The architecture of the system revolves around an intricate deployment pipeline, facilitated by Amazon SageMaker's comprehensive suite of tools for model development, training, and deployment. Additionally, AWS CodePipeline orchestrates the automated release pipeline, ensuring swift and reliable updates. The incorporation of AWS CodeDeploy introduces the blue/green deployment strategy, reducing downtime and mitigating risks by running two identical production environments, enabling seamless transitions between the "blue" and "green" endpoints.

Moreover, the integration of Amazon API Gateway provides a seamless interface for external access to the ML model's functionalities, ensuring secure and efficient communication. This integration ensures robust management and monitoring of the system, enabling swift responses to potential issues and ensuring a seamless user experience.

This report delves into the intricate workings of the NYC TLC Taxi Fare Prediction project, highlighting the significance of integrating diverse AWS services to create a comprehensive and reliable ML deployment pipeline. By focusing on the intricacies of the project's architecture, data preprocessing, model training, and deployment, the report aims to provide a comprehensive understanding of how modern ML and Big Data technologies can be harnessed to derive actionable insights and predictions within the context of the NYC taxi fare ecosystem.

## Related Work:

The paper [Elmi21] presents a novel deep learning architecture called TRES-Net for the prediction of taxi fares by addressing spatial-temporal dependencies. TRES-Net combines Residual Networks (ResNet) for spatial information and Bi-directional Long Short-Term Memory (Bi-LSTM) for temporal information. It constructs a matrix of similar trips based on road network structure and uses a lookup operation to embed spatial features from similar trips. A Bi-LSTM is employed to capture time-series patterns, and the model incorporates a periodically shifted attention mechanism to handle long-term periodic dependencies. TRES-Net also considers external factors like holidays and events, capturing similarity between days of the week and hours of the day. The model combines ResNet and Bi-LSTM, aiming to balance memorization and representation abilities. In terms of performance, TRES-Net outperforms existing models in terms of Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) on real datasets. It achieves better results when compared to nine benchmark methods across various prediction intervals. Future work could focus on real-time model adaptation to changing traffic patterns without retraining from scratch.

This paper [Bagal23] discusses the implementation of a Deep Neural Network (DNN) for predicting taxi and cab fares, focusing on the design and implementation of the algorithm. DNNs are commonly used for image recognition and classification, but they can also be applied to numerical data such as time series data. The paper emphasizes the adaptability of DNNs in handling various data types. The algorithm, termed "Parallel-DNN," is outlined through pseudo-code. It involves steps like feature scaling, splitting data into training and testing sets, and training and testing the DNN model. The dataset used for this project is obtained from Kaggle and contains information related to taxi trips, including fare amounts, timestamps, passenger counts, and location coordinates. The paper highlights the advantages of using DNNs, such as their high accuracy and the ability to handle large datasets with minimal pre-processing.

The results of implementing the DNN are compared to other algorithms, including Linear Regression, Random Forest, Decision Tree, Gradient Boosting, and XGBoost Regressor. The DNN outperforms these algorithms in terms of Root Mean Squared Error (RMSE) and Mean Squared Error (MSE). This paper demonstrates the effectiveness of DNNs in handling numerical data and offers promising results in terms of prediction accuracy. It suggests that DNNs can be valuable tools for enhancing service quality and pricing strategies in the taxi industry.

This paper [Howard18] explores the development of a smart transportation data pipeline using IoT data from the New York City Taxi & Limousine Commission. The authors conduct experiments on various machine learning algorithms, both supervised and unsupervised, to assess their prediction accuracy and computational performance on both commodity computers and distributed systems. Their chosen technologies include Amazon S3, EC2, EMR, MongoDB, Sagemaker and Spark. Notably, the Random Forest Regressors achieve a Root Mean Squared Error (RMSE) of 0.22, a result comparable to Kaggle competition winners. Surprisingly, Logistic Regression outperforms more complex algorithms in terms of training time and accuracy in some instances, underscoring the importance of algorithm selection and data size. The paper's metrics for result measurements include RMSE for prediction accuracy and training times for various machine learning algorithms, particularly comparing Logistic Regression to Random Forest and Gradient Boosted Tree classifiers. This study provides valuable insights into the optimal choice of algorithms and infrastructure for transportation data analysis in distributed systems, making it a pertinent resource for researchers in the field.

This research paper [Ismaeil22] focuses on the application of a decision tree classification model to predict payment types in New York City (NYC) taxi trips. The authors utilize a dataset provided by the NYC Taxi and Limousine Commission, which includes detailed trip records with information on vendors, passenger counts, pickup/drop-off locations, timestamps, and payment types. The goal of the study is to assess the accuracy of payment type prediction using the

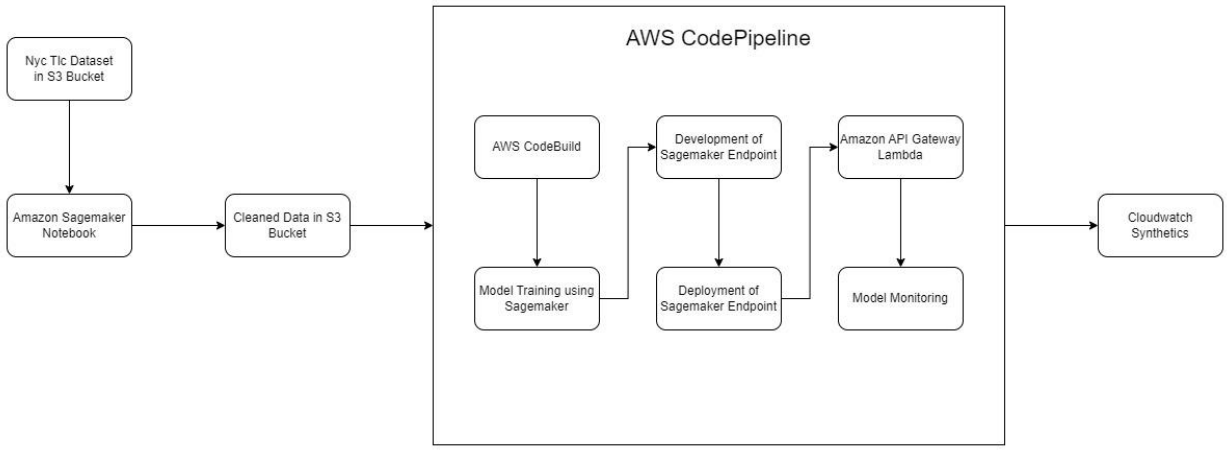
decision tree classification algorithm within the Apache Spark framework.The research demonstrates a high level of accuracy, exceeding 96%, for predicting payment types using the decision tree model. The authors discuss various evaluation metrics, including accuracy, test error, weighted precision, weighted recall, and weighted F-measure. They also investigate the impact of different splitting criteria and node impurity measures (Gini impurity and Entropy) on the model's performance. Overall, this paper offers valuable insights into the application of machine learning, specifically decision tree classification, to real-world transportation data. The findings suggest that this approach can be extended to predict other aspects of taxi trips, such as passenger count and peak-hour surcharges, and potentially be applied to similar car services in different regions.

This research paper [Sun16] analyzes a vast dataset of NYC taxis with Big Data technologies. The research utilizes MapReduce and Hive to understand the patterns and make a prediction on taxi networks. To begin, taxi trip data sourced from the NYC Taxi & Limousine Commission website is transferred into the Hadoop cluster, initiating the data analysis process. This data is then placed within the Hadoop Distributed File System (HDFS), allowing for efficient processing through a suite of tools, including Hadoop MapReduce, Hive, HBase, Pig, and Spark.The primary analysis of the dataset involves MapReduce and Hive, with intermediate results being checked for discrepancies. Subsequently, these intermediate results are processed, confirmed, and graphed to visually represent data relationships and patterns. The architecture treats every taxi as a moving sensor within the NYC taxi system. By combining the historical data gathered from these taxis with a smartphone application, it offers valuable services to taxi riders, drivers, and taxi companies.

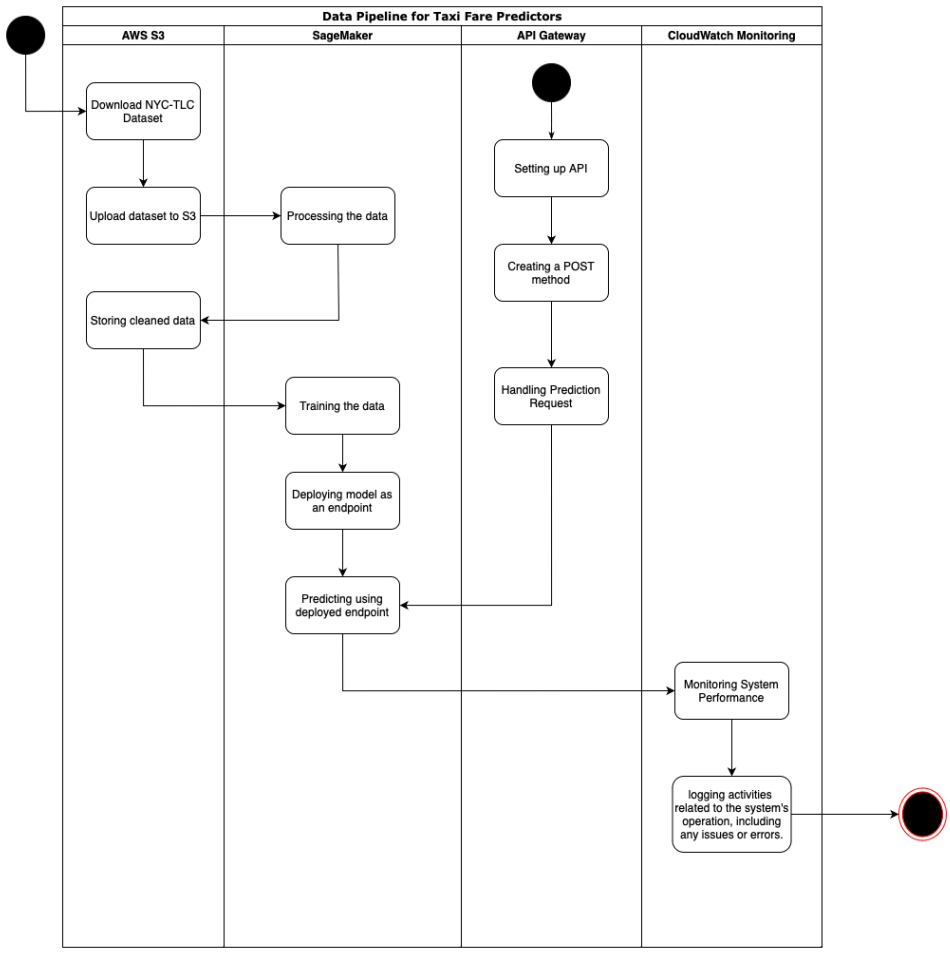
The research paper [Yazici13] utilizes a large dataset of taxi trips to analyze the decision-making process of New York City taxi drivers, aiming to propose strategies for improving access to and passenger satisfaction at John F. Kennedy (JFK) Airport. A one-month subset of taxi GPS data

supplied by the New York City Taxi and Limousine Commission (TLC) is utilized. The dataset comprises essential information, including the latitude and longitude coordinates of pick-up and drop-off locations, along with timestamps. They identified sequential taxi trips and used logistic regression to model whether drivers chose to pick up passengers at the airport or search for customers after each trip. The results of their model confirmed various issues raised by industry stakeholders. Additionally, this methodology can be applied in other locations with access to similar taxi trip datasets.

## Project Design:




To initiate the development of our predictive system, we first upload the converted CSV file containing New York City TLC (Taxi and Limousine Commission) ride details to an S3 bucket. This step allows us to preprocess and construct our prediction model using the dataset.Amazon Sagemaker Notebook plays a crucial role in orchestrating the creation of our predictive system. The initial phase involves preprocessing the dataset, entailing tasks such as feature engineering and eliminating redundant data. Following this, the refined dataset is stored back into the S3 bucket, enabling us to commence the model-building process.The meticulously cleaned data is then returned to the S3 bucket, serving as the starting point for the subsequent stages. AWS CodeBuild comes into play at this juncture, undertaking operations on the dataset and applying the XGBoost algorithm to generate the fare prediction model.Upon the completion of model training, our focus shifts towards the development of a Sagemaker Endpoint, which serves as an essential component of the predictive system. Thorough testing of the Endpoint ensures its readiness for deployment.Once the Endpoint has been thoroughly tested, it is deployed, enabling us to commence the prediction process. Amazon API Gateway handles the incoming prediction requests, ensuring a seamless interaction between the users and the predictive system.Model Monitoring, facilitated through CloudWatch, is employed to oversee the overall performance and health of the system, ensuring its effective functionality and performance.




This activity diagram illustrates the workflow for taxi fare prediction using AWS services. It shows how AWS S3 stores the input data, AWS SageMaker trains the machine learning model, API Gateway handles prediction requests, and AWS CloudWatch monitors system performance and logs activities.

In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system.

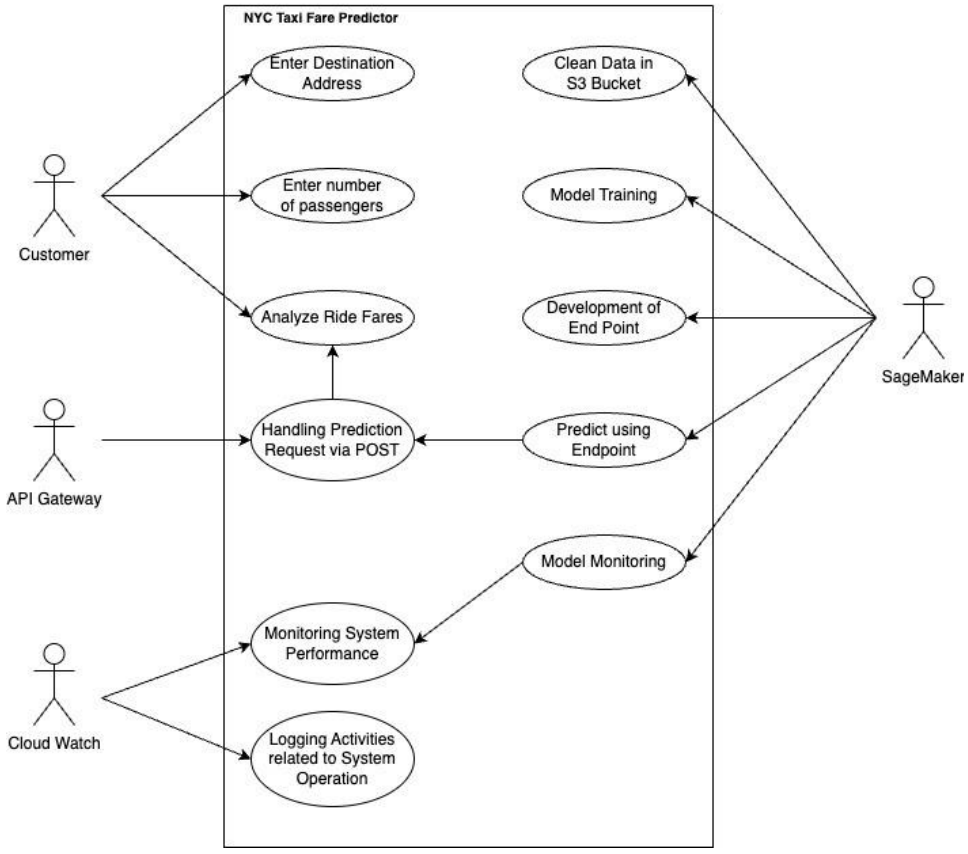
For our project, we have the following actors/system that are responsible for predicting the fare of taxi in New York City:

1) Customer: A customer is the user who is going to use the system to book a cab. His interaction is limited to the front end of the system where he enters his destination, number of passengers and finally selects the cab based on the predicted fare generated by the system.

2) SageMaker: AWS SageMaker does the role of cleaning the data in S3 bucket, training and maintaining the model, developing an endpoint and predicting using that endpoint.

3) API Gateway: API Gateway does the role of handling prediction requests via POST by interacting with the endpoint created by SageMaker and then using the results received to provide options to the user for booking the taxi.

4) CloudWatch : CloudWatch ensures that the system is running healthy by interacting with SageMaker to monitor the model and system performance. It also helps create an overview of the running by actively logging the reports onto the system



## Big Data Challenges:

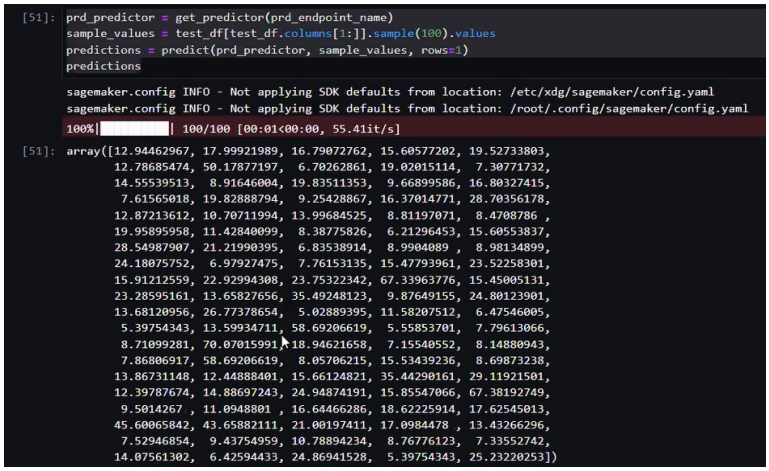
**Storage:** The project involved storing and managing the large dataset, including the training and validation data, in an efficient and scalable manner. Amazon S3 (Simple Storage Service) likely played a significant role in providing a reliable and cost-effective storage solution for the project, allowing it to securely store and retrieve data when needed.

**Complexity:** AWS Step Functions simplify pipeline complexity by offering a visual interface for orchestrating and managing workflow components. By structuring tasks as states within a workflow, Step Functions streamline the coordination of disparate services, easing integration with various AWS resources. This eliminates the need for intricate custom code to manage state transitions, error handling, and retries, thereby reducing the complexity of workflow management. Additionally, the platform's monitoring capabilities enable efficient tracking of workflow execution, aiding in the identification of bottlenecks and simplifying troubleshooting. Ultimately, Step Functions' modular approach and automated handling of scalability and reliability aspects significantly reduce the overall complexity of pipeline orchestration, empowering developers to focus on building and refining individual components rather than navigating intricate workflow intricacies.

**Reliability:** AWS CloudFormation stacks bolster pipeline reliability by enabling infrastructure management through code. These stacks encapsulate sets of resources, ensuring consistent provisioning and updates across environments. By defining infrastructure as code, CloudFormation templates promote reproducibility and version control, mitigating configuration drift and reducing the likelihood of errors during pipeline stages. The automated deployment and rollback capabilities inherent in CloudFormation minimize human-induced mistakes, while dependency management within stacks ensures the orderly provisioning of resources, enhancing the overall reliability of the pipeline by maintaining consistent and controlled environments throughout the development lifecycle.

## Project Outcomes and Reflection:

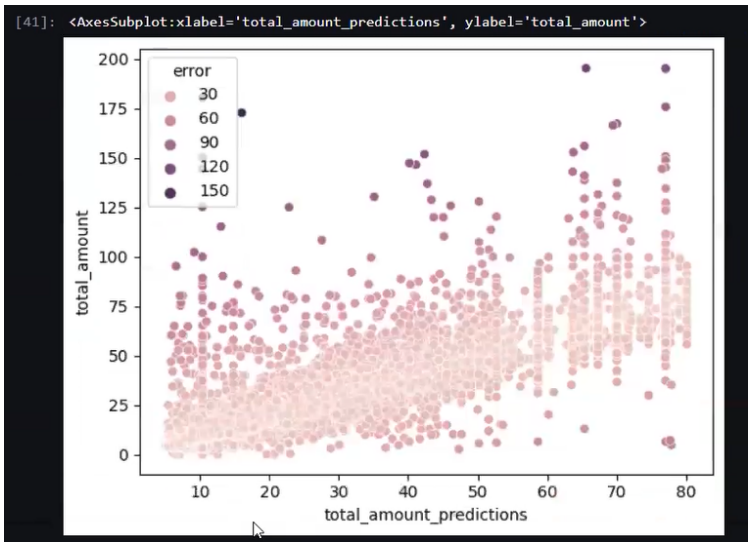
The figure below shows the taxi fare values predicted using the model pipeline created. The model created using XGBoost gives us a better result than the models which used the decision tree model.



```
[51]: prd_predictor = get_predictor(prd_endpoint_name)
      sample_values = test_df[test_df.columns[1:]].sample(100).values
      predictions = predict(prd_predictor, sample_values, rows=1)
      predictions

      sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
      sagemaker.config INFO - Not applying SDK defaults from location: /root/.config/sagemaker/config.yaml
      100%|          | 100/100 [00:01<00:00, 55.41it/s]
[51]: array([12.94462967, 17.99921989, 16.79072762, 15.60577202, 19.52733803,
             12.78685474, 50.17877197,  6.70262861, 19.02015114,  7.30771732,
             14.55539513,  8.91646004, 19.83511353,  9.66899586, 16.80327415,
              7.61565018, 19.82888794,  9.25428867, 16.37014771, 28.70356178,
             12.87213612, 10.70711994, 13.99684525,  8.81197071,  8.4708786 ,
             19.95895958, 11.42840099,  8.38775826,  6.21296453, 15.60553837,
             28.54987907, 21.21990395,  6.83538914,  8.9904089 ,  8.98134899,
             24.18075752,  6.97927475,  7.76153135, 15.47793961, 23.52258301,
             15.91212559, 22.92994308, 23.75322342, 67.33963776, 15.45005131,
             23.28595161, 13.65827656, 35.49248123,  9.87649155, 24.80123901,
             13.68120956, 26.77378654,  5.02889395, 11.58207512,  6.47546005,
              5.39754343, 13.59934711, 58.69206619,  5.55853701,  7.79613066,
              8.71099281, 70.07015991, 18.94621658,  7.15540552,  8.14880943,
              7.86806917, 58.69206619,  8.05706215, 15.53439236,  8.69873238,
             13.86731148, 12.44888401, 15.66124821, 35.44290161, 29.11921501,
             12.39787674, 14.88697243, 24.94874191, 15.85547066, 67.38192749,
              9.5014267 , 11.0948801 , 16.64466286, 18.62225914, 17.62545013,
             45.60065842, 43.65882111, 21.00197411, 17.0984478 , 13.43266296,
              7.52946854,  9.43754959, 10.78894234,  8.76776123,  7.33552742,
             14.07561302,  6.42594433, 24.86941528,  5.39754343, 25.23220253])
```

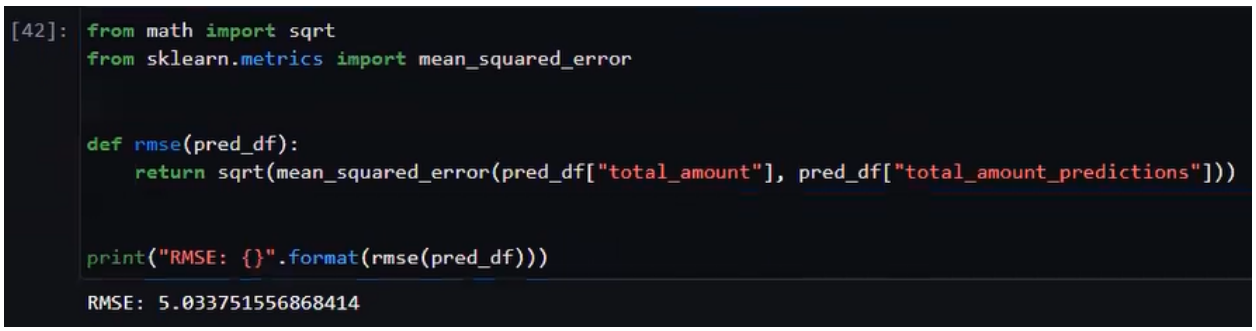
The figure below shows the light pink points which are even lighter than the error 30 point which shows that the error for the prediction is minimal. The outliers which can be seen are due to the hefty amounts of tips given to the cab drivers by passengers which are not considered for predicting the fare amount. For eg. If the cabbie accepts a trip for a short distance then the customer awards a hefty tip to the cab driver. These types of transactions are the outliers that we have.



The image below shows the last input values retrieved from test csv and the output value i.e the amount in dollars which is predicted which is very close to the actual value.

```
Last file:
{
  "captureData": {
    "endpointInput": {
      "observedContentType": "text/csv",
      "mode": "INPUT",
      "data": "18.083333333333332,1.0,2.61",
      "encoding": "CSV"
    },
    "endpointOutput": {
      "observedContentType": "text/csv; charset=utf-8",
      "mode": "OUTPUT",
      "data": "16.790727615356445",
      "encoding": "CSV"
    }
  },
  "eventMetadata": {
    "eventId": "0b412ae9-ffad-42c4-82ef-e773275028a3",
    "inferenceTime": "2023-11-11T04:32:58Z"
  },
  "eventVersion": "0"
}
```

The image below shows the RMSE value of 5.033 which is pretty good with respect to the dataset we are considering. As this type of analysis has not been done this large amount of dataset yet.



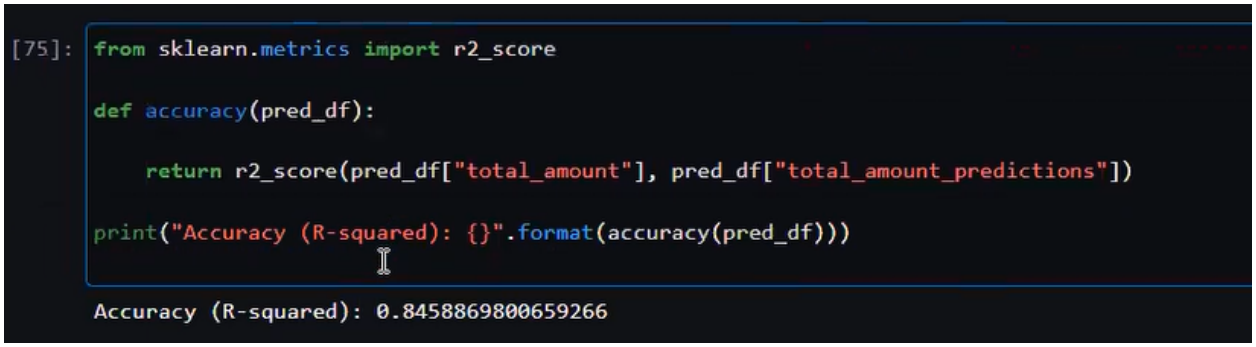
```
[42]: from math import sqrt
      from sklearn.metrics import mean_squared_error


      def rmse(pred_df):
          return sqrt(mean_squared_error(pred_df["total_amount"], pred_df["total_amount_predictions"]))


      print("RMSE: {}".format(rmse(pred_df)))

      RMSE: 5.033751556868414
```

Below is the accuracy that we obtained for our prediction system. We think the accuracy is quite good when considering the complexity of the dataset.



## Conclusion:

The success of the NYC TLC Taxi Fare Prediction project hinges on the seamless integration of cutting-edge technologies and meticulous design of the deployment pipeline. Leveraging Amazon SageMaker's tools for model development, training, and deployment, alongside AWS CodePipeline's automated release pipeline and AWS CodeDeploy's blue/green deployment strategy, ensures a swift, reliable, and low-risk implementation process. The incorporation of Amazon API Gateway enables secure external access, robust management, and efficient issue resolution. This cohesive architecture not only ensures accuracy in predicting taxi fares but also highlights the project's efficiency, reliability, and adaptability to the dynamic landscape of New York City's taxi services. The careful orchestration of these technologies demonstrates a model deployment pipeline that not only functions effectively but also exemplifies the power of modern machine learning and Big Data technologies in addressing real-world challenges.