International Journal of Scientific Research in Science, Engineering and Technology

Print ISSN - 2395-1990
Online ISSN : 2394-4099

OPEN ACCESS

Available Online at : www.ijsrset.com
doi : https://doi.org/10.32628/IJSRSET2411239

Interpretable AI Services for Enhanced Air Quality Forecasting

Ketan Shahapure[1], Samit Shivadekar[1], Bhrigu Bhargava[2]

[1]Department of Computer Science and Electrical Engineering University of Maryland Baltimore County

[2]Student at Harrisburg University of Science and Technology, USA

ARTICLEINFO

Article History:

Accepted: 25 March 2024
Published: 12 April 2024

Publication Issue :
Volume 11, Issue 2
March-April-2024
Page Number :
260-272

ABSTRACT

Most of the Machine Learning (ML) models used these days establish a complex relationship between the in- dependent variables (X) and dependent variable (y). Without understanding the relationship, we risk introducing undesirable features into the predictions. Biased collection of the data, used to build the model, might bolster these undesirable features. The model might soon become unfit for its intended tasks. This project tries to get deeper insights into such black box machine learning models by looking into various ExplainableAI (XAI) tools and provide it as a service to users. These tools when used in conjunction can make complex models easy to understand and operate for the end-user. Specifically, the tools used would help the user of the machine learning model interact with it and monitor how it behaves on changing certain aspects of the data. To facilitate the better understanding of the achieved outcome, this project uses a weather data-set which is used to classify the air quality.

Index Terms — Explainable AI(XAI), Machine Learning, Classifi- Cation, Service Oriented Computing

I. INTRODUCTION

Many industries rely on complex machine learning models to make business decisions, which decide their revenues and affect their customers, products and assets. To cite an example, space vehicle health monitoring is usually driven by large satellite fleet operators. These operators monitor dozens of satellites from a single control center where engineering staff respond to faults and failures if needed. These engineers are also responsible for monitoring the health of the fleet – a task which is now aided by machine learning algorithms. These algorithms are trained using past telemetry data and once trained can detect anomalies. A biased machine learning model could miss a potential displacement of a satellite from its orbit.

. Hence it becomes crucial to understand these models.

AI model not only gives predictions but also explains them. Explanation can be of many types. Feature importance, rules, feature contribution are some examples of explanation which you will come across. Explainable AI is broadly classified

into two types: Global explanation and local explanation.

1) Global Explanations: Global explanation provides a high-level overview of the model. Often times, a model built on certain assumptions by one data scientist is passed to another data scientist not necessarily attached with the assumptions. The receiver data scientist will have a tough time understanding the model before using it. In such a case an overall behavior of the model can be extremely helpful in saving the time of receiver data scientist in understanding the model. A sample global explanation can provide the variation of output of the model with respect to a feature. Some of the global explanation techniques which I would be discussing are Eli5 feature weights, SHAP, Accumulated local effects and Partial dependency plots.
2) Local Explanations: Where global explanation takes as input the entire data and outputs the overall construct of the model, a local explanation technique works on a single instance. This technique predicts/estimates the output of the instance given to it along with an explanation. The explanation usually shows contribution of various features towards the prediction. Some of the local explanation techniques discussed in this paper are LIME and Anchors.

II. RELATED WORK

A lot of literature is out there which deals with explainable AI as a technique. Some of these techniques are discussed in detailed in section V. Rulex[3] and Dalex[4] are two services which deal with XAI. However Explainable AI as a service for Air Quality has not been done before. So, this project proposes a novel idea to provide users with explained air quality predictions.

III. THE DATA SET

The data-set chosen for this task is the Air Pollution Prediction and Forecasting using Air Quality Index. The snippet below shows the part of the data with it's features.

TABLE I

SNIPPET OF THE DATA-SET

Temp	Windspeed	Pressure	NO2	Rain	PM10	AQI	PM25
14.033	14.1197	1015.13	17.4	56.8	87	168	89.1
15.056	14.2646	1015.63	7	56.8	122	177	105.5
15.916	3.9284	1015.94	28.5	56.8	95	174	100.2
16.094	14.1036	1016.41	14.7	56.8	79	169	89.6
16.094	11.0446	1016.51	7.5	56.8	63	162	76.3

A. Data-Set Details

This data-set[https://github.com/grtvishnu/Air-Pollution-Prediction-and-Forecasting/tree/master/Data] has 7288 data entries.All data values are numerical here and are much better to handle. It contains features such as;

- Temperature (°C)
- Wind Speed (Km/h)
- Pressure (Pa)
- NO2 (ppm)
- Rainfall (Cm)
- PM10 (g/m3)
- PM2.5 (g/m3)
- AQI (Air Quality Index)

So, from the above features, the AQI feature was eliminated since the model we are using will predict it. To convert it into a classification task, we assign ranges to the Air quality index in the training data-set and thus, creating five classes of AQI. The classes were derived from airnov.gov [1] and are as follows.

Daily AQI Color	Levels of Concern	Values of Index	Description of Air Quality
Green	Good	0 to 50	Air quality is satisfactory, and air pollution poses little or no risk.
Yellow	Moderate	51 to 100	Air quality is acceptable. However, there may be a risk for some people, particularly those who are unusually sensitive to air pollution.
Orange	Unhealthy for Sensitive Groups	101 to 150	Members of sensitive groups may experience health effects. The general public is less likely to be affected.
Red	Unhealthy	151 to 200	Some members of the general public may experience health effects; members of sensitive groups may experience more serious health effects.
Purple	Very Unhealthy	201 to 300	Health alert: The risk of health effects is increased for everyone.
Maroon	Hazardous	301 and higher	Health warning of emergency conditions: everyone is more likely to be affected.

Fig. 1. AQI Severity levels

B. Data-set partitioning

We tested our model's predictions for the test data-set with the actual values of AQI corresponding to the test data set and achieved an accuracy of 97.5% This project uses a classification model.

IV. MODEL -XGBOOST

- XGBoost is an optimized distributed gradient boost- ing library designed to be highly efficient, flexible and portable.
- The library is laser focused on computational speed and model performance.
- Speed: A study performed benchmarks Random Forest Implementations with various libraries and XGBoost shows the fastest results.
- Performance: It is the go-to library algorithm (gradient boosting decision tree) for competition winners on the Kaggle competitive data science platform. It is also the best suited library for structured data.
- Some key features of XGBoost's Gradient Boost:
  - Gradient Boosting algorithm also called gradient boosting machine including the learning rate.
  - Stochastic Gradient Boosting with sub-sampling at the row, column and column per split levels.
  - Regularized Gradient Boosting with both L1 and L2 regularization.[2]

V. XAI: TOOLS AND BACKGROUND

A. Global Explanations

SHAP: SHAP [5] is a famous XAI related research work. SHAP stands for SHapley Additive exPLanations. It is a post-hoc model explanation technique which uses a surrogate model to find Shapley values [6] of all the features. Shapley values were given by Lloyd Shapley for game theory. In a multi-player game setting, multiple players work together to complete a task. Not all players contribute equally to the task. In order to reward each player, it is essential to compute the contribution of each player. The contribution of each player according to Lloyd Shapley is equivalent to finding Shapley value for that player.

Calculating Shapley values φ: In a coalition game with a set N (having n players) and a function v that maps subsets of players to real numbers : $v:2^N \rightarrow R$, with $v(\varphi)=0$, where φ denotes an empty set. The function is called characteristics function. The function v is inter- preted as follows: if S is a coalition of players, then v(S) is called the worth of coalition S. It describes the total expected sum of payoffs the members of S can obtain by co-operation. The Shapley value is a way to distribute total gains to the players. The amount that player i gets given a multiplayer game (v,N) is,

$$\varphi(v) = \sum_{S \subseteq N \atop i} \frac{|S|!\,(n-|S|-1)!}{n!} (v(S \cup \{i\}) - v(S))$$ Here n is the total number of players and the sum extends over all subsets S of N not containing player i. The way to interpret this: Imagine the coalition being formed one player at a time, with each player demanding their contribution $v(S \cup i) - v(S)$ as a fair compensation and then for each actor take the average of this contribution over all possible permutations in which the coalition can be formed [7].

- Shapley values in machine learning context: In case of a machine learning model, imagine getting a prediction from the model to be a game where multiple features are players working towards calculating predictions. The contribution of each feature can be the Shapley value of that feature. Similar to what we discussed earlier Shapley value of each feature is calculated as the average marginal contribution of that feature considering all pos-sible permutations of coalitions of features not including the feature of interest. Now let us

consider a simple machine learning model, to simplify the discussion so far:

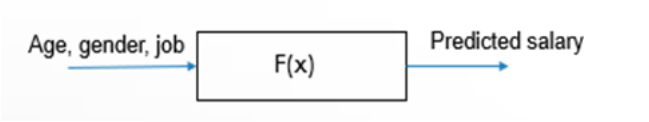


Figure 2 shows a simple machine learning model which considers features like 'Age', 'Gender' and 'Job' to predict salary of a individual. Given a predicted salary let us find the contribution of feature Age. Each box in the Figure 3 is a model with feature/features value of random instance x0 given to it mentioned inside it along with prediction of it. All possible permutation of coalitions of the tree features are considered in the Figure 3. The weights connecting the models which do not contain the feature Age to those models which do contain the feature

Age are highlighted in red.

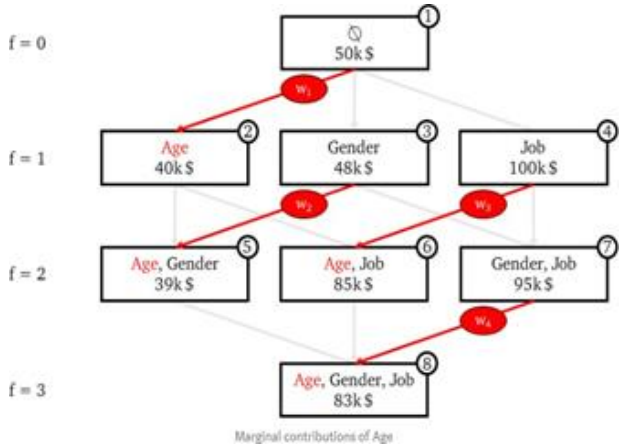


Fig. 3. Simple model

Considering the models labeled '1' and '2' in the Figure 15, marginal contribution of *Age* is the difference in prediction of the two models, i.e. $MC_{Age,Age}(x0)$=40- 50=-10. Similarly marginal contributions of *Age* with respect to pairs of model '3' and '5', '4' and '6', and

'7' and '8' are calculated.

$$MC_{Age,\{Age\}}(x0) = 40 - 50 = -10k\$$$

$$MC_{Age,\{Age.Job\}}(x0) = 85 - 100 = -15k\$$$

$$MC_{Age,\{Age.Gender,Job\}}(x0) = 83 - 95 = -12k\$ \quad (1)$$

The Shapley value of feature *Age* is given by

$$SHAP_{Age}(x0) = w1\,MC_{Age,\{Age\}}(x0) + w2\,MC_{Age,\{Age,Gender\}}(x0) + w3\,MC_{Age,\{Age,Job\}}(x0) + w24\,MC_{Age,\{Age,Gender,job\}}(x0) \quad (2)$$

In order to calculate the weights, the following rules are used:

- Sum of all the weights which connect models having feature of interest with models excluding it is 1, i.e. $w_1 + w_2 + w_3 + w_4 = 1$
- The sum of the weights of all the marginal contributions to 1-feature-models should equal the sum of the weights of all the marginal contributions to 2-feature-models and so on. In other words, the sum of all the weights on the same "row" should equal the sum of all the weights on any other "row". In our example, this implies $w_1 = w_2 + w_3 = w_4$
- All the weights of marginal contributions to f-feature-models should be equal, for each f. In other words, all the edges on the same "row" should have equal weights. In our example, this means $w_2 = w_3$. Therefore, on solving $w_1 = \frac{1}{3}$ $w_2 = \frac{1}{6}$ $w_3 = \frac{1}{6}$ $w_4 = \frac{1}{3}$ In general, for a model with F features in total, the weigh for an edge connecting f-1 feature model to f feature model can be given by $\frac{1}{f.\binom{F}{f}}$.

On simplifying equation for $SHAP_{Age}(x0)$ and substi- tuting values for weights we get:

$$SHAP_{Age}(x_0) = 1.\binom{3}{1}^{-1} \times MC_{Age,\{Age\}}(x_0) + 2.\binom{3}{2}^{-1} \times MC_{Age,\{Age,Gender\}}(x_0) + 3.\binom{3}{3}^{-1} \times MC_{Age,\{Age.Gender,Job\}}(x_0) \quad (3)$$

Similarly,

$$SHAP_{Age}(x_0) = -11.33k\$$$

$$SHAP_{Gender}(x_0) = -2.33k\$$$

$$SHAP_{job}(x_0) = -46.66k\$ \quad (4)$$

Adding up all the above gives +33$, which is exactly the difference output of the full model 83k$ and dummy model with no features 50k$. This leads us to an important property of SHAP which is 'adding the SHAP values of all features of an instance gives the difference between the prediction of the model and the expected value of the model. Hence the name SHapley Additive explanations. The outputs related to SHAP shown in the section D, are obtained from a python library called SHAP itself. This library approximates a way to compute these Shapley values by utilizing a surrogate model. A concise discussion around this is presented below.

- Calculation of Shapley values from the python library SHAP:

The way we calculated Shapley values in the simple model discussed above, you can notice the number of times we took predictions of the model. For a model with F features in total, the model should be called 2^F times. This becomes impractical for a large value of $F(>50)$. Hence a surrogate linear model is trained to approximate the actual black box model, since linear models are faster to train and give faster predictions.

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j \quad (5)$$

$g$is the explanation model, z' is a coalition vector in dicative of the features included for a prediction. A 0 in the vector z' indicates exclusion of a feature and 1 indicates inclusion. Steps in which ϕ are calculated are listed below: SHAP compute these Shapley values of all features for an instance x following the below steps (refer to Figure 4):

- Step 1 is to sample coalitions zl ∈ {0, 1}M, k ∈ 1,….K indicates feature present.
- Step 2 is to get prediction of each zl by first converting the zl k to original feature space and then applying the model f (hx (zl))
- Compute the weight of each zl using SHAP kernel.
- Fit weighted linear model. When I say fitting, think of the plain old machine learning linear regression problem of fitting where you are learning these coefficients (Shapley values in our case).
- Return the Shapley values ϕ k, the coefficients from the linear model.

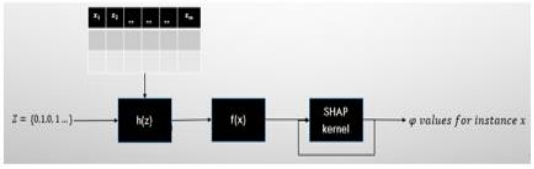


Fig. 4. SHAP Python library

For a multi class problem SHAP shows feature importance for each class as shown in the Figure 4. Since the black box model fed to SHAP library was trained on 7 features. Figure 5 shows the importance of those7 features. The plot is trivial to interpret. For instance, the feature PM 25 contributes most towards the class 0 evident from the predominant blue color.

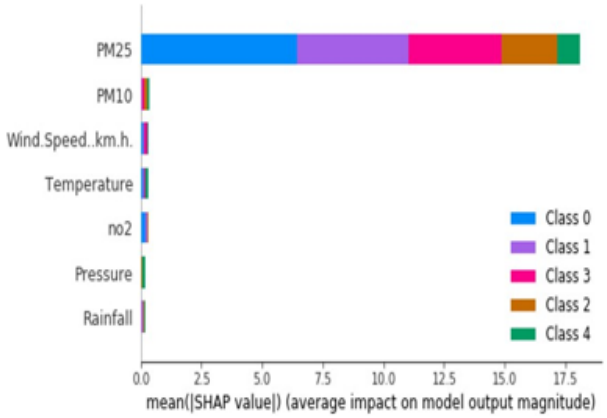


Fig. 5. SHAP Feature Importance

Partial Dependence plots: Partial dependency plots show the variation of model output with a feature value. When applied to a linear regression model these plots always show a linear variation. The

equation for partial dependence function for a feature xs is given by the equation 6.

$$\hat{f}_{x_s}(xs) = E_{x_c}\ \hat{f}(xs, xc) = \int_{x_c} \hat{f}(xs, xc)\, dP(xc) \quad (6)$$

The x_s are the features for which the partial dependence function should be plotted and x_c are the other features used in the machine learning model f^ Usually, there are only one or two features in the set S. The feature s in S are those for which we want to know the effect on the prediction. The feature vectors x_s and x_c combined make up the total feature space X. Partial dependence works by marginalizing the machine learning model output over the distribution of the features in set C, so that the function shows the relationship between the features in set S we are interested in and the predicted outcome. By marginalizing over the other features, we get a function that depends only on features in S, interactions with other features included x`.

The estimated value of f^x_s (x_s) is usually estimated by calculating averages in the training data.

$$\hat{f}_{x_s}(x_S) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}(x_S, x_C^i) \quad (7)$$

In the formula above, x(i) are actual feature values from the data for the features in which we are not interested, n is the number of instances in the data. PDP assumes the feature in set S are not correlated to the features in set C. If this assumption is violated, the averages calculated for the partial dependence plot will include data points that are very unlikely or even impossible. For classification-based machine learning problems, where the output of the model are probabilities, the partial dependence plots display the probability of a certain class given different values of features in S. Let us consider a simple data with three features and three data points to better understand. Tabular representation of this data is shown in Table 2:

TABLE II

SIMPLE DATA-SET

A	B	C	Y
A1	B1	C1	Y1
A2	B2	C2	Y2
A3	B3	C3	Y3

In order to find the partial dependency of feature A, we marginalize all the other features. Table 3 shows the data we get after marginalization. Each value of feature A is listed with every combination of values of features B, C. Each data point from the marginalized data is given to the model to get its output, shown in the column Y of Table 3. Partial dependence of A at A=A1, would be given by mean of all the outputs where A1 value occurs. The mean values are shown in the last column of Table 3. Table 4 shows the final data which is plotted to obtain the partial dependence of A.

TABLE III

MARGINALIZED DATA

A	B	C	Y	Mean
A1	B1	C1	Y11	
A1	B2	C2	Y21	
A1	B3	C3	Y31	Y(A1)
A2	B1	C1	Y12	
A2	B2	C2	Y22	
A2	B3	C3	Y32	Y(A2)
A3	B1	C1	Y13	
A3	B2	C2	Y23	
A3	B3	C3	Y33	Y(A3)

TABLE IV

PARTIAL DEPENDENCE OF FEATURE A

A	A1	A2	A3
Y	Y(A1)	Y(A2)	Y(A3)

Notice that there can be a value of feature A, which will never occur with certain values of features B, C, highlighting the drawback of Partial dependence plots. This drawback is overcome by Accumulated local effects discussed in the next section.

Accumulated Local Effects: Like partial dependence plots accumulated local effects show variation of output of the model with a feature of set of features. In case of classification machine learning models

which give probabilities of different classes, these plots show the variation of probability of the class with feature or set of features. These plots overcome the drawback with partial dependence plots when the features are correlated. Unlike partial dependence plots which marginalize other feature values and average their prediction, accumulated local effects considers conditional distribution of other feature values and averages the difference in predictions. This section provides details of calculation. The formal equation of ALE is shown in equation 8

$$f_{x_s,ALE}(x_s) = \int_{z_{0,1}}^{x_s} E_{X_c|x_s}\left[\hat{f}^s(X_s, X_c) \mid X_S = z_S\right] dz_s - constant = \int_{z_{0,1}}^{x_s}\int_{x_c} \hat{f}^s(z_s, x_c)\, P(x_c|z_s)\, dx_c dz_s - constant \quad (8)$$

Compare this to equation 6. You will notice that the average is taken over difference in predictions shown by the gradient dz_S. All these local differences in the predictions are accu- mulated over the range of features in S shown in equation 9 by th additional integral $\int_{z_{0,1}}^{x_S}$. For actual computation, the z's are replaced by grid of intervals over which we compute the change in prediction. Instead of directly averaging the prediction the ALE calculated the difference in prediction conditional on the feature S and integrates the differences over same feature S. The interval difference isolates the effect of the feature of interest and blocks the effect of correlated features [8]. Notice a constant is subtracted from the final value to center the ALE plot so that average effect over the area is zero.

Estimation of ALE: Figure 6 shows the division of feature of interest, into partitions such that partition ends up with certain data points. Equation 12 best summarizes the estimation of ALE.

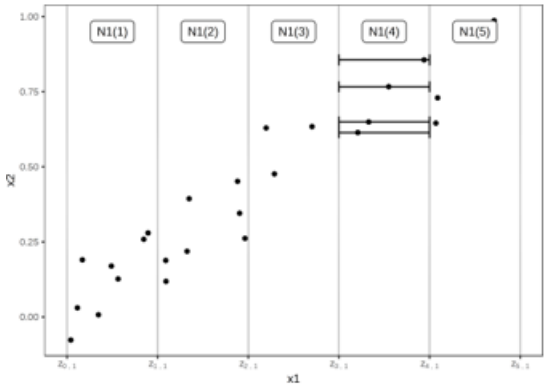


Fig. 6.

$$f_{j,ALE}(x) = \sum_{k_j(x)} \frac{1}{n_j(k)} \sum_{i:x_j^{(i)} \in N_j(k)} \left[f\left(z_{k,j}, x_{\setminus j}^{(i)}\right) - f\left(z_{k-1,j}, x_{\setminus j}^{(i)}\right) \right] \quad (9)$$

The difference within the square brackets in equation 12 is the effect the feature has on the model for an individual instance in a certain interval. The first summation from the right add us all these local effects in the neighborhood $N_j(k)$. This sum is divided by the number of instances in the neighborhood $N_j(k)$, to obtain the average difference of predictions for this interval. This average in the interval is covered by the term local in the name ALE. The second summation from the right is to accumulate the average effects across all the intervals. The ALE of a feature value that lies for example in the third interval is the sum of the effects of first, second and third intervals. The word accumulated in ALE reflects this.

Figure 7 shows a sample ALE plot for the feature.

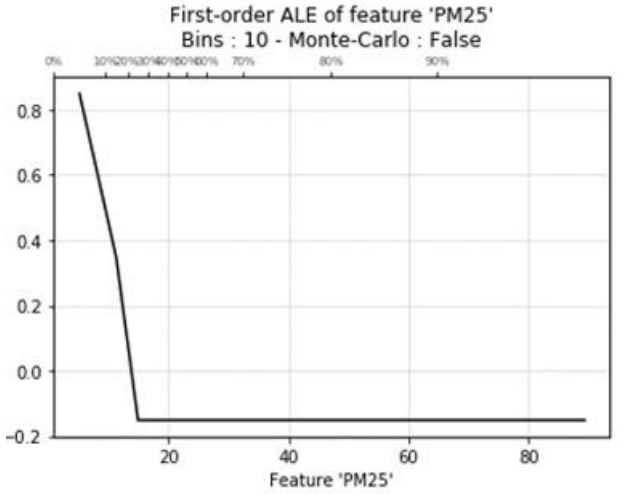


Fig. 7. ALE of PM 2.5 for Class 0

B. Local Explanation

LIME: Local explanation techniques produce explanation for a single instance. LIME [6] is one such technique. LIME [5] stands for Local Interpretable Model Agnostic Ex- planation. This technique uses a local surrogate model to fit/approximate a prediction of the black box model. Given a single instance and its prediction from a complex black box model, LIME tests what happens to the prediction on variation of feature values of the given instance. To that effect LIME creates a sample data set around the instance of interest and gets output of each instance of this sample data set from the black box model. On this new data set an interpretable machine learning model is trained, which is weighted by the proximity of sampled instances to the instance of interest.

The trained model only needs to approximate the local prediction of the selected instance.

- Select an instance you desire an explanation on.
- Generate sample data points around the selected in- stance.
- Weigh the new samples according to their proximity to the instance of interest.
- Train a simple linear model on the generated data set.
- Return the weights of the simple trained model.

An example LIME output is shown in Figure 8. Each column in the figure below shows the feature contribution with respect to a class which that column represents. For instance, the feature 'PM25' contributes a value of -0.398 towards the probability of the instance belonging to class 0.

Fig. 8. LIME

Anchor: Anchors [6] is a useful local explanation tech- nique which creates rules around feature values. These rules anchor the prediction, in a sense that the model will give the same prediction as long as these rules are satisfied irrespective of changes to any feature values. Like LIME, a sample data is created around instance interest. However, instead of surrogate models used by LIME, the resulting explanations are expressed as easy to understand IF-THEN rules called anchors. Anchors have a notion of coverage and precision. Coverage refers to the proportion of sample instances, predictions of which the anchor holds, and precision implies the extent to which the rules in the anchor are exclusively responsible for the predicted outcome.

Formal definition of an Anchor A is given by equation 10.

$$E_{D_x(z|A)}\left[1_{f(x)=f(z)}\right] \geq \tau,\ A(x) = 1 \qquad (10)$$

Where: x is the instance of interest A is the set of predicates collectively forming an anchor such that A (x) = 1 when all the predicates in anchor A correspond to x^1 s feature values. f denotes the

classification model we need explanation on. Dx(.|A) indicates the distribution of neighbours of x, matching

A. $0 \leq \tau \leq 1$ denotes the precision threshold. This ensures

only those rules are a part of anchor which predict the same output with a probability of at least τ.

Finding Anchors: Calling a complex model for each data point from the created sample is an impractical idea. There- fore, the authors proposed an approximate method to compute

these anchors, by introducing a precision parameter δ, such that $0 \leq \delta \leq 1$. The samples are drawn until a statistical confidence is reached with the rules. Equation 11 states this.

$$P(prec(A) \geq \tau) \geq 1 - \delta, \text{ where } prec(A) = E_{D,(z|A)}\left[1_{f(x)=f(z)}\right] \quad (11)$$

Equation 10 and 11 are combined and extended by the notion of coverage. Coverage is the anchor's probability of applying to its neighbours.

$$cov(A) = E_{D_{(z)}}[A(z)] \quad (12)$$

Rationale is to maximize this coverage simultaneously satis- fying equation 11.

$$\max_{As.t.P\,(prec(A)\geq\tau)\geq 1-\delta} cov(A) \quad (13)$$

Aim is to find, out of all the rules, those rules which satisfy the precision threshold given the probabilistic confidence. If an anchor uses all the rules of the data, it becomes more precise, but coverage reduces. Conversely, if the anchor uses less rules the precision reduces but coverage increases. Hence, there is a trade-off between. More the number of features in the data, more specific the anchors are. There exists a python library, named anchor which implements this technique. The code uses reinforcement learning and beam search algorithm to the best anchor. Discussion of how the code works falls outside the scope of this project.

Figure 9 shows screenshot of output from anchor library, when applied to our model.

Fig 9. Anchor

What-if tool: This tool is really useful to make Explainable AI interactive. It enables a user to select any instance of interest, observe the output of the model, and make changes to any feature value of the instance to monitor the change in the output of the model. Figure 10 shows a screenshot of what-if tool we developed.

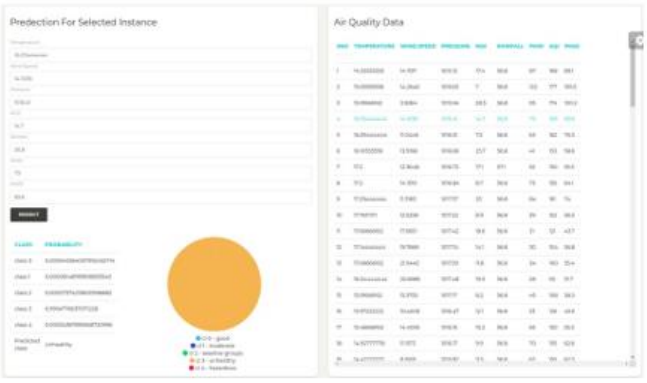


Fig 10. Anchor

VI. XAI AS A SERVICE: ARCHITECTURE

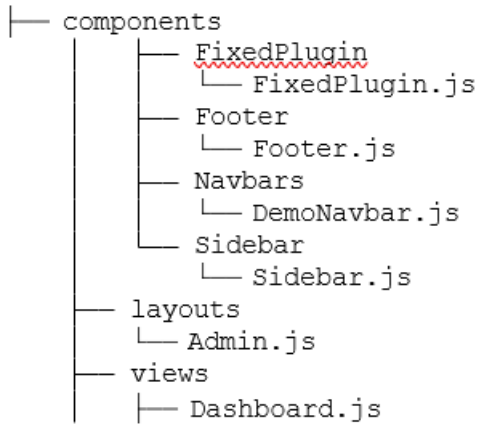
Following is the technology stack that was selected for the development of XAI as a service.

- React – frontend
- Django – backend
- NodeJS
- AJAX

Programming Languages

- Javascript
- Python

The component hierarchy is as follows:



The top-level component that renders as the user lands on the application admin page is Admin.js. Within its body, the Admin.js component calls other child components in order. It first invokes the Sidebar component that is responsible to display the side navigation bar. Next, it calls the Navbar component to render the navigation panel at the top of the display. The most critical component to the application the Dashboard component is then called followed by the footer component. The best thing about this structure is we have separate components for each part of the UI. So for the subsequent screens, we have the liberty to customize. For the application, we use react routing, everything on the screen including the sidebar, navigation, and footer is fixed, and the component that lies in the middle of all keeps on getting updated based on the react route that has been selected from the side panel giving the user a seamless shift from one component to other, sort of like a single page application.

Dashboard Component This component is the core of the application. It enables the support to connect to the backend. A play button appears on the dashboard upon clicking this button an AJAX API call is made to the backend, the component fetches data, parses it, and displays it on the UI. After a part of the application has been rendered, the next API call made for the What If? the feature is done using AJAX, as we retrieve the results using the fetch API call we ensure that previously fetched results are not refreshed and only the specific part of the page is reloaded not the entire screen.

Architecture: The application uses two different servers and enables interaction between them. The react code is deployed on the Node server. And all the Django code is deployed on a different server that works on a different port.

The machine learning model is trained with the pre=processed air quality data-set and is all set to give out results. And the code to output all the XAI plots/explanations and the code that is run for giving out specific prediction is embedded within Django and deployed onto the server. We make use of Django's Model-View-Controller design architecture to our advantage. The python code on Django is wrapped inside functions that act as an API endpoint for the request coming from React. Depending on where the user clicks on the UI, react makes an AJAX call using the fetch API that is inbuilt in react, this API call invokes a URL in urls.py that in turn invokes the wrapper functions. The Django code exposes itself to the react API calls, in other words, we have designed a RESTFUL API using Django that can be used to give out XAI explanations in JSON format. This makes the frontend and the backend loosely coupled and encourages code reusability. We are strong believers of open-source and this design decision was made to make that any other application on the web can utilize the Machine Learning code that we have done, if we release the API, the endpoints can be used by any developer on the web to fetch the XAI explanations.

The results generated by the XAI techniques that are implemented using python are parsed into JSON. And these results are sent to the React server as a response to the API call that was made. React parses the JSON data, and populates it on the UI. Similarly, for the 'What if?' feature, as the user clicks on the

prediction button an AJAX API request is made to the Django server, the request captures the values that the user wants prediction on, and hits the API endpoint, the python code extracts the prediction based on the values fetched from React, gives an output that is again sent via JSON. We choose JSON for data exchange because it is a widely accepted industry format.

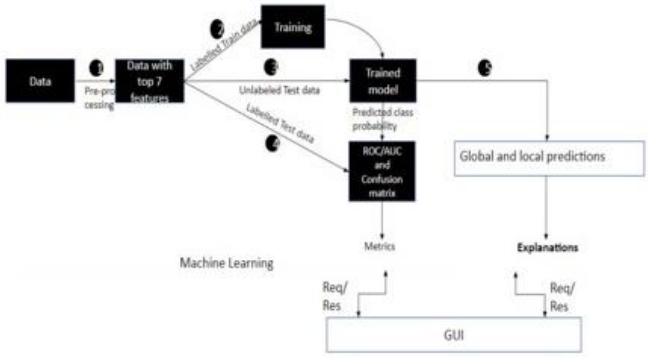


Fig 11. Service Architecture

The front-end code is modularized to encourage code reusability, collaboration, and readability. Each module maps to a react component. The components are hierarchically structured, and the data passing between each component is achieved using react props.

The figure 11 briefly illustrates the development process for the service.

Confusion matrix:

```
[[ 356    0    0    0    2]
 [   0 1195    0    0    0]
 [   0    0  242    0    1]
 [   0    0    0  321    3]
 [   1    3    5    0   58]]
```

	precision	recall	f1-score	support
0	1.00	0.99	1.00	358
1	1.00	1.00	1.00	1195
2	0.98	1.00	0.99	243
3	1.00	0.99	1.00	324
4	0.91	0.87	0.89	67
accuracy			0.99	2187
macro avg	0.98	0.97	0.97	2187
weighted avg	0.99	0.99	0.99	2187

Fig 12. Classification Report

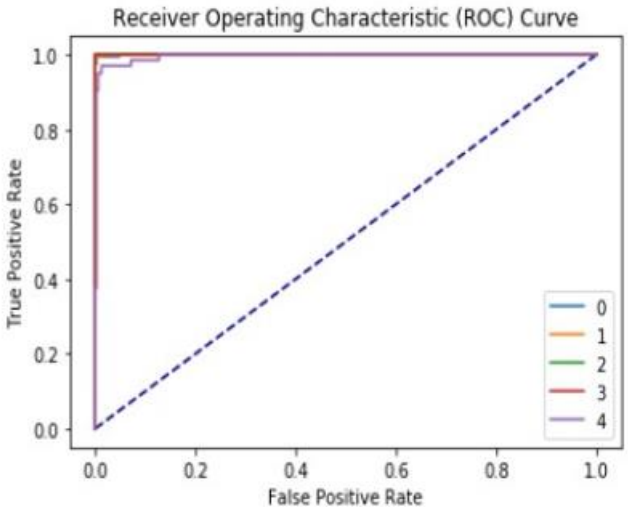


Fig 13. AUC-ROC curve

VII. RESULTS

A. Black Box results

Figure 12 and 13 shows the overall performance of the trained model. Figure 12 shows the Confusion matrix and Classification report of the model. We can infer the model performs good overall with the confusion matrix diagonally dominant. The AUC-ROC curves in the Figure 13 also are close to ideal behaviour. However, these plots give little information about the model.

B. Explained model

To simplify our analysis let us select an instance (Figure 14) from our data set. Notice the value of PM25 = 64.1. The present prediction and Eli5 output is shown in Figure 15 and Figure 16. The selected instance is pointing towards Class 3 air quality with a .99 probability. Also notice the high contribution of PM25 towards class 3.

8	17.2	14.1519	1016.84	8.7	56.8	73	155	64.1

Fig 14. Selected instance

Fig 15. Class probability for selected instance when PM2.5 = 64.1

Fig 16. Eli5 explanation for selected instance when PM2.5 = 64.1

If we modify the value of PM25 from 64.1 to a high value of 20000, we notice the class of the selected instance changes to Class 1 as shown in figure 17. Also notice the change in contribution of PM25 in figure 18 for Class 0 and Class 3.

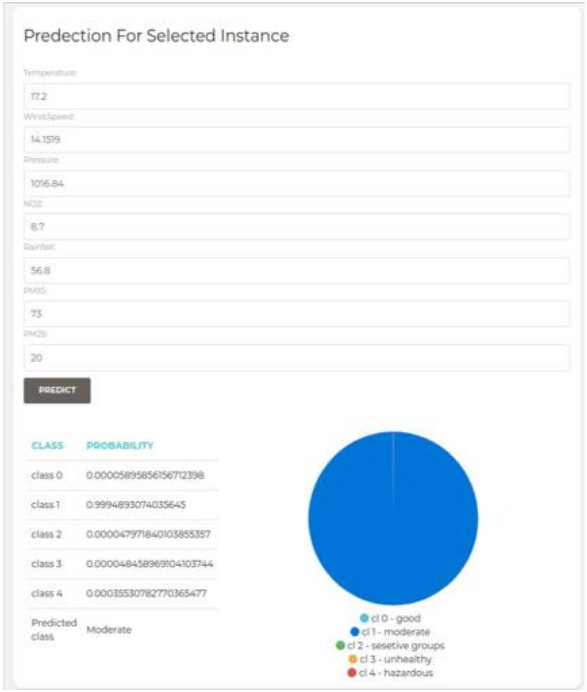


Fig 17. Class probability for selected instance when PM2.5 = 20000

Fig 18. Eli5 explanation for selected instance when PM2.5 = 20000

A comprehensive explanation could be derived for each in- stance using the components of the front end, hence improving the interpretation of the model.

VIII. CONCLUSION

XAI is a topic under extensive research. As with any new technology, there is an ocean of knowledge yet to be researched. Work on explanation of Neural networks is also topic under development. This project has scraped just the surface. We have learnt the different approaches that have been developed

and tried to corroborate results from them to develop a simple yet effective Explainable AI system. Not only has this project strengthened our knowledge on Machine Learning in general but has also given us a deeper understanding of models like Random Forest and XGBoost. We are now more enthusiastic about going deeper into the concepts we have learnt and try to contribute in whatever means possible.

FUTURE DIRECTION

Applying Explainable AI to time series data and creating monitors which keep a track of changes in the data and in the explanations is something which can be looked into. We have tried our best to achieve the goal of Explaining the Explainable AI, by using different approaches. But at the end of the day the user would still have to observe the results from these explainable techniques to notice a pattern. To help in making his/her task easier, we can look to develop a story or simple human readable paragraph or sentences for each explanation. Additionally, more research on anchor methods would be really helpful to decide the trade-off between coverage and precision.

REFERENCES

[1] Phil Dickerson, AirNow Program Director (EPA/OAQPS) John E. White,AQI and AirNow 101: How AirNow Works,2018 NAQ Confer-nce, Austin TX

[2] Chen, T., Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). New York, NY, USA: ACM. https://doi.org/10.1145/2939672.2939785

[3] https://www.rulex.ai/

[4] http://dalex.drwhy.ai/

[5] A Unified Approach to Interpreting Model Predictions, Scott Lundberg and Su-In Lee, 2017.

[6] Shapley, Lloyd S. (August 21, 1951). "Notes on the n-Person Game – II: The Value of an n-Person Game" (PDF). Santa Monica, Calif.: RAND Corporation.

[7] https://towardsdatascience.com/shap-explained-the-way-i-wish- someone-explained-it-to-me-ab81cc69ef30

[8] Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models, Daniel W. Apley and Jingyu Zhu,2019.

[9] Why Should I Trust You?": Explaining the Predictions of Any Classifier, Marco Tulio Ribeiro and Sameer Singh and Carlos Guestrin, 2016.

[10] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. An- chors: High-precision model-agnostic explanations. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)