Scuola Normale Superiore

Classe di Scienze

SCUOLA
NORMALE
SUPERIORE

Beijing Air Quality Prediction and Classification

Intorduction to Machine Learning

Friday 12th December, 2025

Author: Giulio Cordova

giulio.cordova@sns.it

Contents

1 Data Understanding

1.1 Feature description

The *Beijing* $PM_{2.5}$ dataset includes 43824 entries and 13 variables, covering temporal, meteorological features, and $PM_{2.5}$ concentration, i.e. the concentration (expressed in grams per cubic meter) of fine particulate pollution with a diameter less than 2.5 µm. In detail, the dataset contain the following features, divided into three macro classes:

- **Temporal:**
  - `year`: integer ranging from 2010 to 2014. Indicates the year the data was recorded.
  - `month`: integer ranging from 1 to 12. Indicates the month the data was recorded.
  - `day`: integer ranging from 1 to 31. Indicates the day of the month that the data was recorded.
  - `hour`: integer ranging from 0 to 23. Indicates the time of day at which the data was recorded.
- **Meteorological:**
  - `DEWP`: integer ranging from -40 and 28. Indicates the dew point, expressed in degree celsius.
  - `TEMP`: integer ranging from -19 to 42. Indicates the temperature, expressed in degree celsius.
  - `PRES`: integer ranging from 991 to 1046. Indicates the atmospheric pressure, expressed in hPa.
  - `cbwd`: categorical variable with 4 possible values: NW (Northwest), NE (Northeast), SE (Southeast), and cv (Calm variable). Indicates the combined wind direction.
  - `Iws`: float ranging from 0.45 to 585.6. Indicates the cumulated wind speed, expressed in m/s.
  - `Is`: integer ranging from 0 to 27. Indicates the cumulated hours of snow.
  - `Ir`: integer ranging from 0 to 36. Indicates the cumulated hours of rain.
- **Target variable:**
  - `pm2.5`: float ranging from 0 to 994. Indicates the $PM_{2.5}$ concentration, expressed in $\mu g/m^3$.

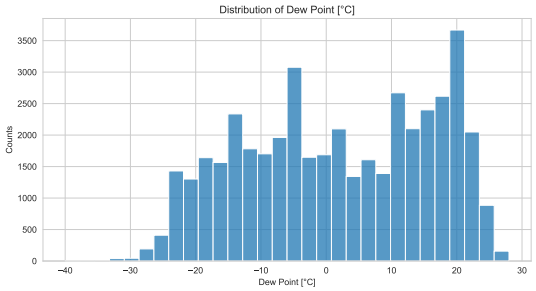
In the dataset, there is also a `No` column, which is just an index and does not provide any useful information. Therefore, this column has been removed during the data preparation phase.

The temporal variables have been used to create a new variable, `datetime`, which combines year, month, day, and hour into a single timestamp. This new variable has been set as the index of the DataFrame, facilitating time series analysis and visualization. It was checked that there were no missing timestamps in the dataset, confirming that the data is continuous over the recorded period.

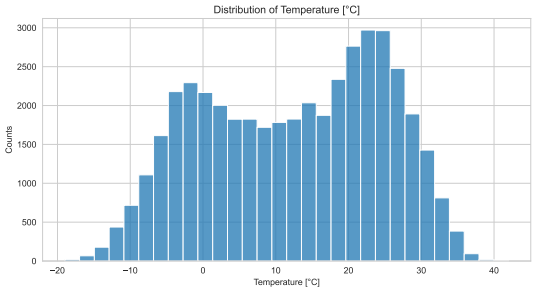
The meteorological variables' distributions are shown in Figure 1. These distributions can be used to check for any anomalies or outliers in the data, as well as to understand the general weather conditions during the data collection period. All the values of the meteorological variables fall within reasonable ranges (i.e. there are no values that are physically not possible), indicating that the data is reliable and suitable for further analysis.

The categorical variable `cbwd` presents only four unique values, as shown in Figure 2. The distribution of these categories indicates the prevailing wind directions during the data collection period. The 'SE' (Southeast) direction is the most frequent, followed by 'NW' (Northwest), 'cv' (Calm and variable), and 'NE' (Northeast).

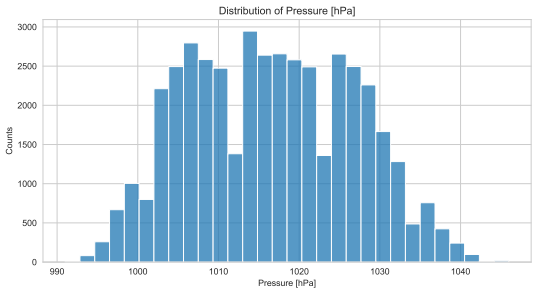
The target variable, `pm2.5`, has a distribution that is right-skewed, as shown in Figure 3a. Most of the $PM_{2.5}$ values are concentrated at the lower end of the scale, with a long tail extending towards



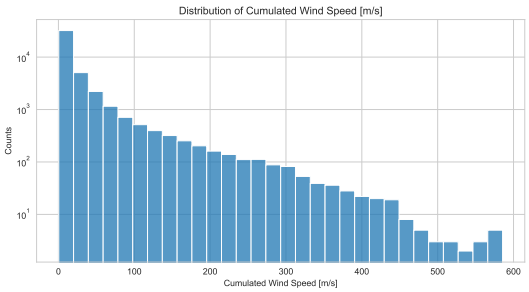
(a) DEWP distribution



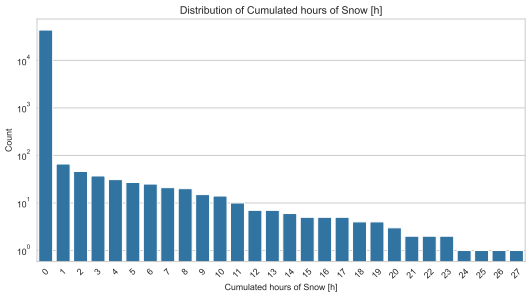
(b) Temperature distribution



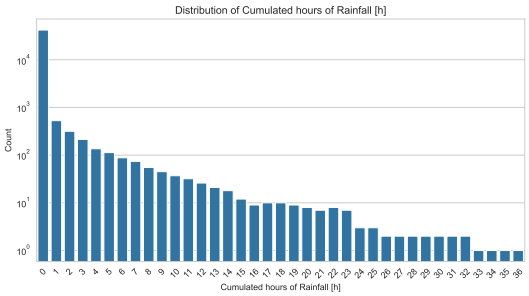
(c) Pressure distribution



(d) Cumulated wind speed distribution



(e) Cumulated hours of snow distribution



(f) Cumulated hours of rain distribution

Figure 1: Distributions of the meteorological variables in the Beijing $PM_{2.5}$ dataset. Each subplot shows the frequency of records for each value of the respective meteorological variable. The distributions exhibit various shapes, indicating diverse weather conditions in the dataset.

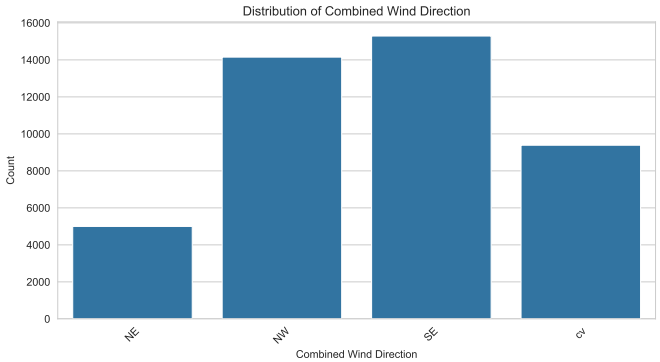
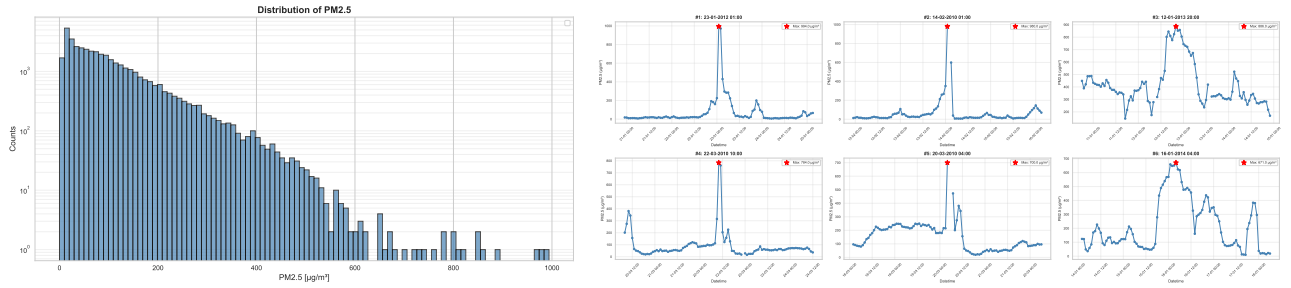


Figure 2: Distribution of the `cbwd` categorical variable in the Beijing $PM_{2.5}$ dataset. The bar chart shows the frequency of each wind direction category, with 'SE' being the most common, followed by 'NW', 'cv', and 'NE'.



(a) Distribution of the `pm2.5` target variable in the Beijing $PM_{2.5}$ dataset. The histogram shows a right-skewed distribution, with most values concentrated at the lower end and a long tail extending towards higher $PM_{2.5}$ concentrations. There are few very high pollution levels, indicating occasional extreme air quality events.

(b) Time series of $PM_{2.5}$ values over 100 hours around the 6 highest recorded values in the Beijing $PM_{2.5}$ dataset. The plots show significant pollution spikes, indicating that these extreme values are genuine observations rather than data errors.

Figure 3: $PM_{2.5}$ distribution and extreme values analysis

higher values. This indicates that while low $PM_{2.5}$ concentrations are the most common, higher pollution levels do occur, albeit less frequently.

In the histogram, there are some extreme values that could be considered outliers. To inspect them, I plotted the time series of the $PM_{2.5}$ values over 100 hours around the 6 highest recorded values, as shown in Figure 3b. The plot reveals the temporal shape of these extreme events, and it is possible to categorize them in two classes:

- **Isolated spikes:** These are characterized by a sudden and sharp increase in $PM_{2.5}$ values, followed by a rapid decrease back to lower levels. The spikes last for a short duration, 1 or 2 hours. This pattern suggests a brief pollution event, possibly due to a transient source of pollution. For example, the spikes of January 23rd, 2012 and February 14th, 2010, happen at midnight, suggesting it was the use of fireworks for the new year celebrations that caused these sudden increases in pollution (dates retrieved with wikipedia[4]). On the other hand, the high pollution events of march 2010 are due to a drought and sand storm that affected Beijing during that period[2].

- **Sustained high pollution periods:** These show a gradual increase in $PM_{2.5}$ values, reaching a peak and then slowly decreasing over time. This pattern indicates a more prolonged pollution event, likely influenced by persistent environmental factors or continuous pollution sources.

These extreme values are part of significant pollution spikes, suggesting that they are genuine observations rather than data errors. Therefore, these outliers have been retained in the dataset for further analysis.

1.2 Missing values

The dataset contains missing values only in the target variable `pm2.5`. A total of 2066 missing entries are present, accounting for approximately 4.7% of the entire dataset. A first visualization of the missing values over time is shown in Figure 4. Here, the red lines indicate the timestamps where the $PM_{2.5}$ values are missing. From a first glance at this plot, the missing values appear to be randomly distributed over the entire time span of the dataset, without any obvious patterns or clusters.

A more detailed analysis of the variables associated with the missing $PM_{2.5}$ values is presented in Figure 5. The distributions of year, month, day, and hour for the missing values are shown in subplots (a) to (d). These distributions reveal that the missing values are not uniformly distributed across the temporal dimensions, with earlier years in the data aquisition period showing a higher frequency

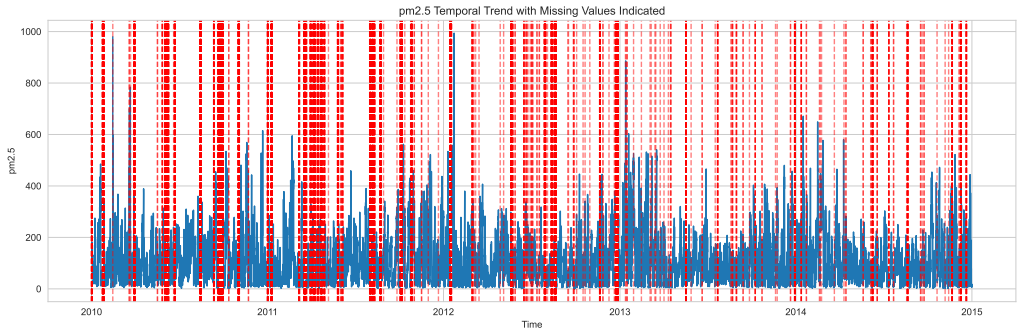
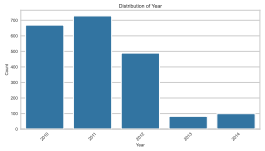
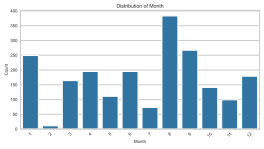


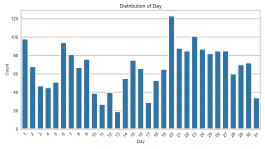
Figure 4: Missing values of $PM_{2.5}$ over time in the Beijing $PM_{2.5}$ dataset. The red lines indicate the timestamps where the $PM_{2.5}$ values are missing. The missing values appear to be randomly distributed over the entire time span of the dataset, without any obvious patterns or clusters.



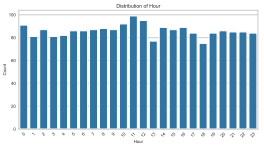
(a) Year distribution



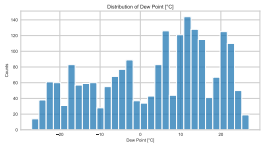
(b) Month distribution



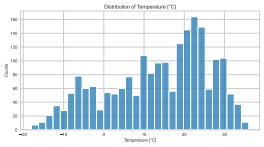
(c) Day distribution



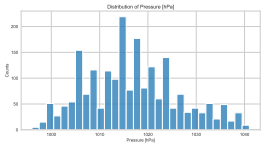
(d) Hour distribution



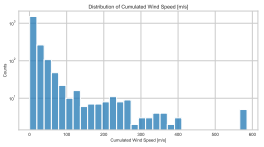
(e) DEWP distribution



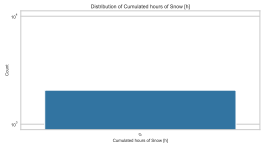
(f) Temp distribution



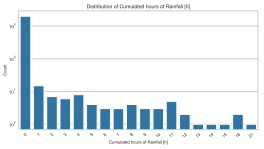
(g) Pressure distribution



(h) Iws distribution



(i) Is distribution



(j) Ir distribution

Figure 5: Distributions of the temporal and meteorological variables of the missing values in the Beijing $PM_{2.5}$ dataset. Each subplot shows the frequency of records for each value of the respective variable.

of missing entries. The hour distribution appears more uniform, suggesting that the missing values are not concentrated at specific times of the day. The month and day distributions also show some variability, indicating that certain months and days have a higher incidence of missing $PM_{2.5}$ values. For example, august and september have a higher number of missing values, while feabruary has the least.

Also the meteorological variables associated with the missing $PM_{2.5}$ values have been analyzed, as shown in Figure 5. The distributions of DEWP, TEMP, PRES, Iws, Is, and Ir for the missing values are presented in subplots 5e to 5j. These distributions reveal that the meteorological conditions does not really have a significant impact on the occurrence of missing $PM_{2.5}$ values, with no significant pattern indicating a deviation from the original ones (Figure 1). The only exception is the cumulated hours of snow (Is), which has only values equal to zero for the missing $PM_{2.5}$ entries. I have no clear interpretation for this fact.

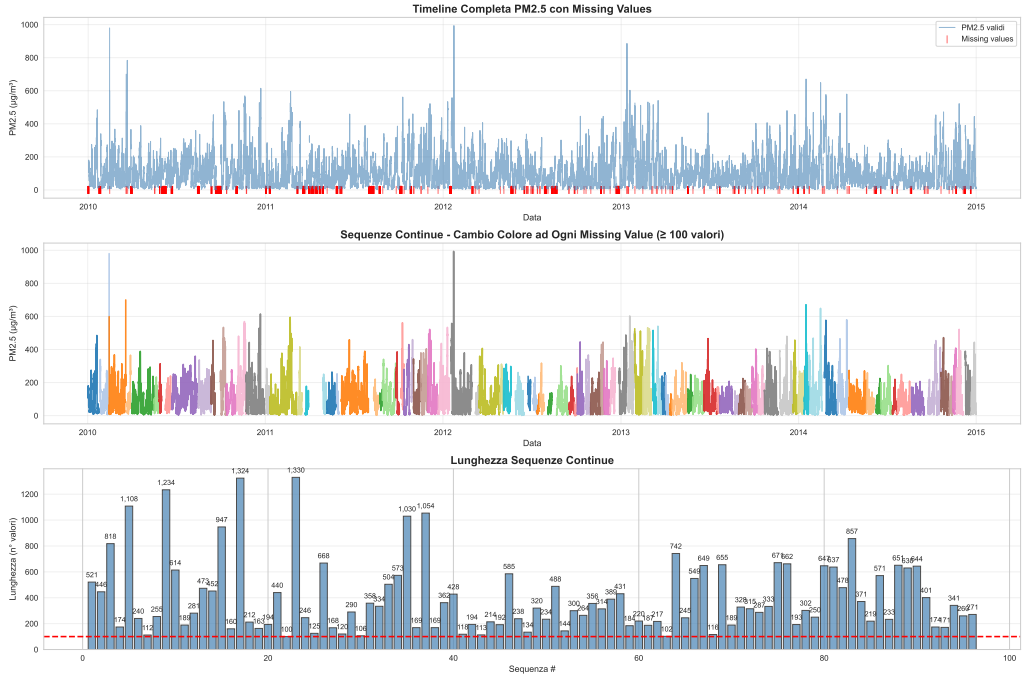


Figure 6: Data preparation for LSTM sequences: dataset is divided into continuous segments without missing $PM_{2.5}$ values, and sequences of at least 100 entries are created from these segments for LSTM input.

2 Main task: Prediction

The aim of this task is to predict the value of $PM_{2.5}$, In order to so, I employed a Long Short-Term Memory (LSTM) neural network, which is well-suited for time series prediction tasks due to its ability to capture long-term dependencies in sequential data.

The network consists of an input layer, one or more LSTM layers, and a dense output layer.

Since we want to infer the value of $PM_{2.5}$ using its predecessors, only this variable was used, and the others were disregarded.

2.1 Data preparation

The input of the network consists of sequences of the past 100 $PM_{2.5}$ values, while the output is the predicted $PM_{2.5}$ value for the next time step. The first step in this task was to prepare the data for the architecture that we defined: since we are dealing with a time series prediction problem, it is necessary to create sequences of past $PM_{2.5}$ values to use as input for the LSTM network. To do this, I created overlapping sequences of length 100 from the $PM_{2.5}$ time series, windowing the original data. Each sequence consists of 100 consecutive $PM_{2.5}$ values, and the corresponding target value is the $PM_{2.5}$ value immediately following the end of the sequence.

This task was not banal given the presence of missing values in the $PM_{2.5}$ time series. If we simply created sequences without considering the missing values, we would end up with sequences that contain NaN values, which cannot be used as input for the LSTM network. If the missing values were removed, the temporal continuity of the data would be lost, which is crucial for time series prediction tasks. To handle this, I implemented a strategy to create sequences of at least 100 entries that only include complete data, i.e., sequences that do not contain any missing $PM_{2.5}$ values. Figure 6 illustrates this data preparation process. In the top panel, we see the original $PM_{2.5}$ time series with missing values indicated by red lines. In the center panel, the continuous segments of the time series (i.e., segments without missing values) of at least 100 entries are highlighted, each with a different color. Finally, in the bottom panel, we see the length of each continuous segment.

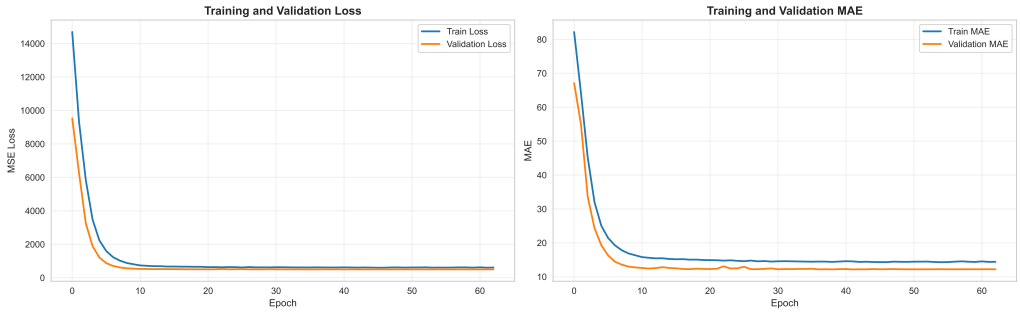


Figure 7: Training history

Once the sequences are obtained, the data is windowed to create chunks of exactly 100 entries, which are then used as input for the LSTM network. The dataset is then split into training, validation, and test sets in a randomic way. One thing that is often done in time series prediction tasks is to split the data chronologically, using the earlier part of the time series for training and the later part for testing. However, in this case, I opted for a random split to ensure that the training, validation, and test sets are representative of the entire dataset, given that the data spans multiple years and seasons. This approach helps to mitigate potential biases that could arise from temporal trends in the data, given also the fact that the Beijing government implemented several measures to reduce air pollution starting from 2013 (i.e. right in the middle of our dataset).

I also ensured that there was a correct proportional amount of 100-long sequences in the training and test dataset (70% training, 30% test), and that there was no overlap between the sequences in the training and test sets, to prevent data leakage.

In the end, 94 continuous sequences were found, from which 20,010 100-long chunks were extracted for the training set (which represent 69.3% of the total number of chunks found), while the testing dataset contains 8,861 windows.

Before giving the data to the LSTM network, I normalized the $PM_{2.5}$ values using standard scaling, which scales the values to have zero mean and unit variance.

2.2 Model

The LSTM network was implemented using the Keras library in Python. The architecture consists of two LSTM layers, followed by a dense output layer with a single neuron to predict the $PM_{2.5}$ value. In between the LSTM layers, dropout regularization was applied to prevent overfitting. The network was trained using the Adam optimizer and mean squared error (MSE) as the loss function.

In order to identify the best hyperparameter configuration, a study with `Optuna` library was conducted. The hyperparameters that were optimized are the following:

- Number of LSTM layers (1 or 2)
- Number of units in each LSTM layer (from 32 to 256, step 32)
- Dropout rate (0.0 to 0.5)
- Learning rate (1e-5 to 1e-2)
- Batch size (32, 64, 128, 256)

In the end, the best hyperparameter configuration found is the following:

- 1 LSTM layers
- 64 units in the LSTM layer

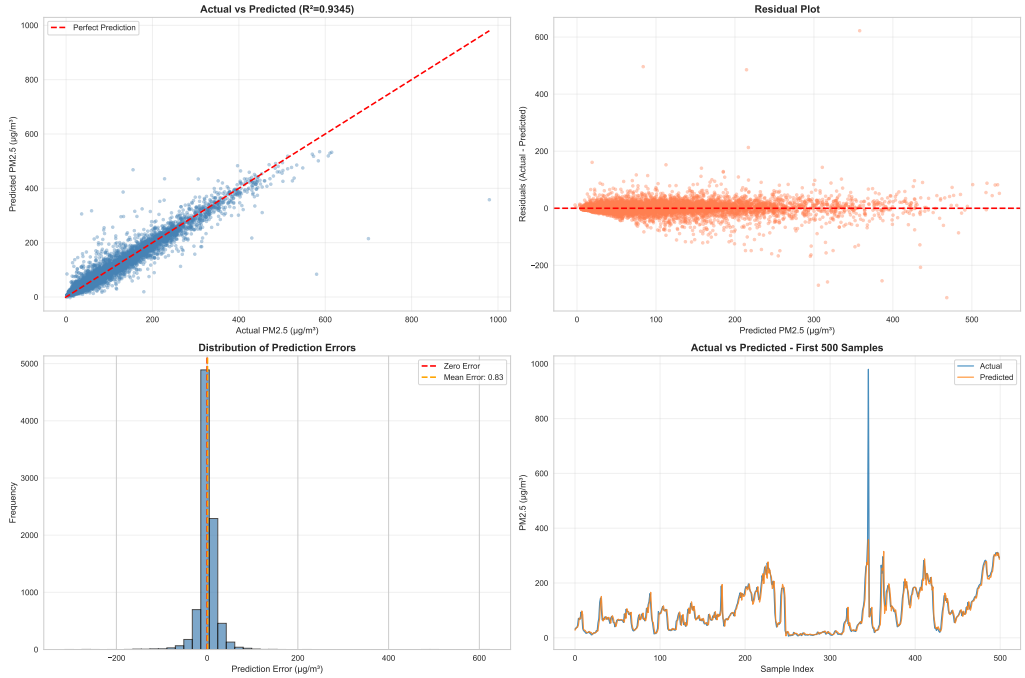


Figure 8: Test predictions

- Dropout rate of 0.1
- Learning rate of 0.0004
- Batch size of 64

These hyperparameters were used to train the final LSTM model on the training set, with a validation split of 20% of the training data. Furthermore, two callbacks were used during training:

- EarlyStopping, to stop training if the validation loss does not improve for 15 consecutive epochs,
- ReduceLROnPlateau, to reduce the learning rate by a factor of 0.5 if the validation loss does not improve for 10 consecutive epochs.

The model was trained for a maximum of 200 epochs, but the training stopped after 63 epochs due to early stopping.

The training history of the model is shown in Figure 7, which displays the training and validation loss over epochs, on the left hand side, and the mean absolute error (MAE) on the right hand side.

2.3 Results

Figure 8 shows some summary plots about the predictions of the trained LSTM model on the test set.

In the top left panel, there is a scatter plot of the true $PM_{2.5}$ values versus the predicted values, with the ideal y=x line shown in red. The points are closely clustered around the y=x line, indicating that the model's predictions are generally accurate. There a few points that deviate significantly from the line, indicating some prediction errors, especially at higher $PM_{2.5}$ values. The density of points is higher at lower $PM_{2.5}$ values, which is consistent with the distribution of $PM_{2.5}$ in the dataset. Furthermore, a saturation effect is visible at high $PM_{2.5}$ values, where the model tends to underpredict the true values. This saturation begins roughly around 450 $\mu g/m^3$, but the effect is hard to quantify, in order to the limited statistics in the higher region. The limited number of high $PM_{2.5}$ samples in the training data could also be a cause of this saturation: the model struggles to learn accurate predictions in this range.

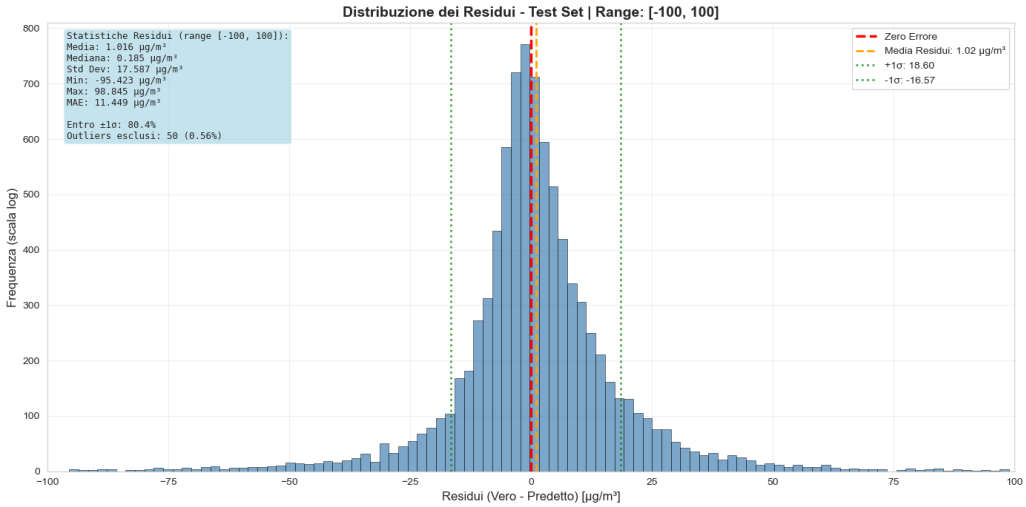


Figure 9: Zoom on residuals histogram in the range -100 to 100 $\mu g/m^3$

The top right panel shows the residuals (i.e., the difference between the true and predicted $PM_{2.5}$ values) as a function of the true $PM_{2.5}$ values. This plot helps to visualize any systematic biases in the model's predictions across the range of $PM_{2.5}$ values, and one can see that there are no systematic effects, except for the saturation at high $PM_{2.5}$ values already described. Furthermore, from this plot the scale of the residuals can be appreciated: most of them fall within the range of -100 to 100 $\mu g/m^3$, with some outliers extending beyond this range (e.g. -300 or 600)

The bottom left panel shows the distribution of the residuals, which appears to be roughly centered around zero, indicating that the model does not have a significant bias in its predictions. The distribution differs significantly from a Gaussian shape, having instead heavier tails, indicating that there are more extreme prediction errors than would be expected in a normal distribution.

Finally, the bottom right panel shows the time series of the true and predicted $PM_{2.5}$ values for a subset of the test set. From this plot, we can see that the model is able to capture the overall trends and fluctuations in the $PM_{2.5}$ values over time, although there are some discrepancies between the true and predicted values at certain points, especially for high values of $PM_{2.5}$.

Considering the whole test set, the model achieves a MAE of 12.33 $\mu g/m^3$, a Root Mean Squared Error (RMSE) of 23.02 $\mu g/m^3$, and a coefficient of determination (R^2) of 0.9345. The high R-squared value indicates that the model explains a significant portion of the variance in the $PM_{2.5}$ values, demonstrating its effectiveness in capturing the underlying patterns in the data. The RMSE indicates the statistical bias in the dataset, which may appear high, but let's consider the vast range of $PM_{2.5}$ values in the dataset (from 0 to almost 1000 $\mu g/m^3$).

To perform a better evaluation, Figure 9 depicts a zoom of the residuls histogram in the range -100 to 100 $\mu g/m^3$. The distribution is centered around zero, and from the statistics (in the top left corner) we can see that the mean of the residuals is 1.06 $\mu g/m^3$, which corresponds to the statistical bias in the estimate when outliers are removed. The median of the distribution is 0.185 $\mu g/m^3$, indicating that the slight underprediction bias is quite low in this range.

2.4 Filling the missing value gaps in the original dataset

Given the good prediction capabilities, I decided to use this model to fill the gaps in the original $PM_{2.5}$ time series. To do this, I employed an iterative approach, where the model predicts one missing value at a time, using the most recent 100 known or previously predicted $PM_{2.5}$ values as input.

Figure 10 shows the resluts of this approach, with the following color coding:

- blue: original $PM_{2.5}$ values.

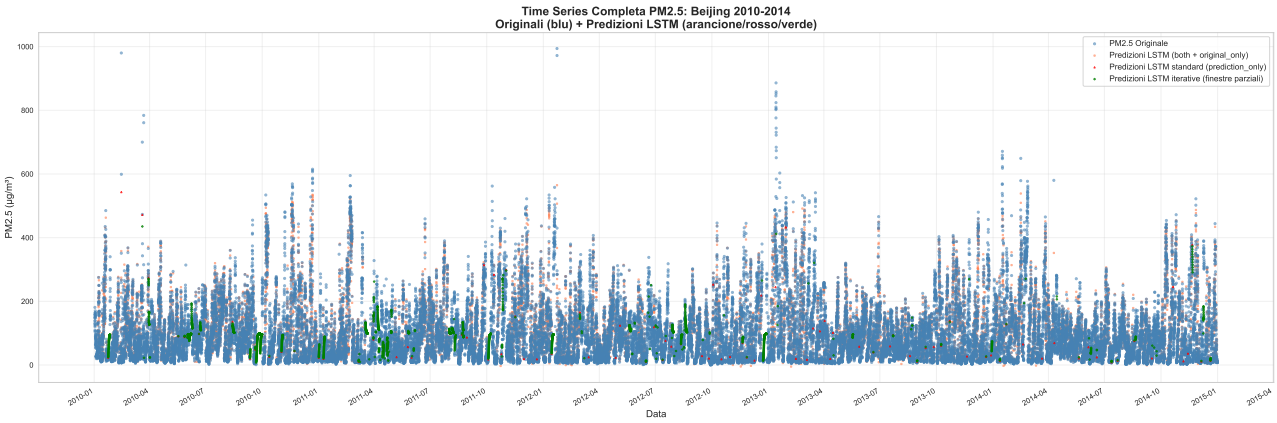


Figure 10: Full predicted time series

Table 1: Best hyperparameters for the 3 different tested models

Model	Units	Dropout	Learning rate	Batch size	Number of layers
GRU	64	0.15	0.001	32	3
RNN	64	0.13	0.0005	64	2
LSTM	128	0.01	0.0005	64	1

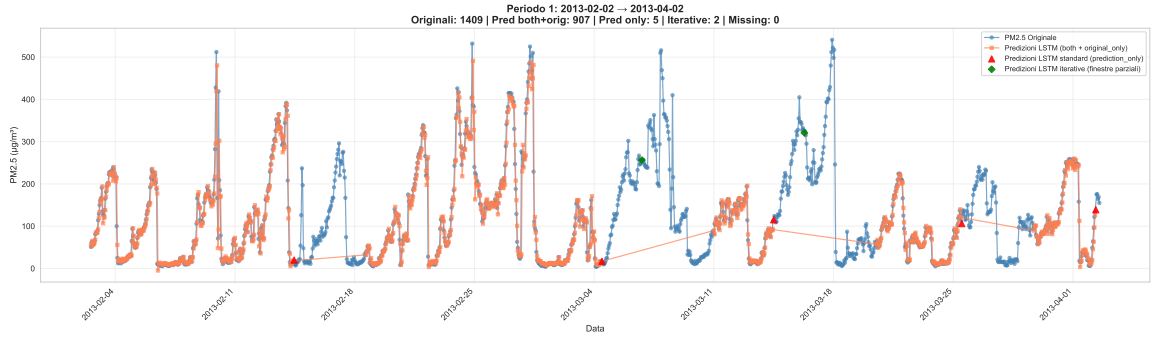
- orange: predicted $PM_{2.5}$ values using 100 known original values as input. These points are the same ones used to train, validate and test the model, since they have a correspondance in the original values.
- red: missing values that have 100 known predecessors. These values are predicted only, since there is no reference to compare them in the original dataset.
- green: filled gaps with the iterative approach. In the 100 predecessors used to predict these values, there is at least one missing point that was already predicted.

Since the whole time span is a bit difficult to visualize, I randomely selected 4 periods containing 2-months-long time series. The randomly selected periods are shown in Figure 11, where the same color coding of Figure 10 is used. In these periods, it is visible the high prediction capability of the model, even when using previously predicted values as input. Red values are very close to the blue ones. This is a good indication, since it is expected that usually there are no sudden jumps that lasts only 1 or 2 hours, but a rather continuity in the data is expected. Regarding green values (predicted values using previously predicted data as input) is a bit more difficult to say. Isolated values fall within the ballpark of the original values, but when there are long sequences of missing values, the predictions tend to "flatten" and do not capture the variability of the original data as well. An example of this is visible in period 3 (Figure 11c) and period 4 (Figure 11d), where a long sequence of missing values is present around May 25th 2011 and 17 March 2011 I would expect to see more oscillations, as in the original dataset.

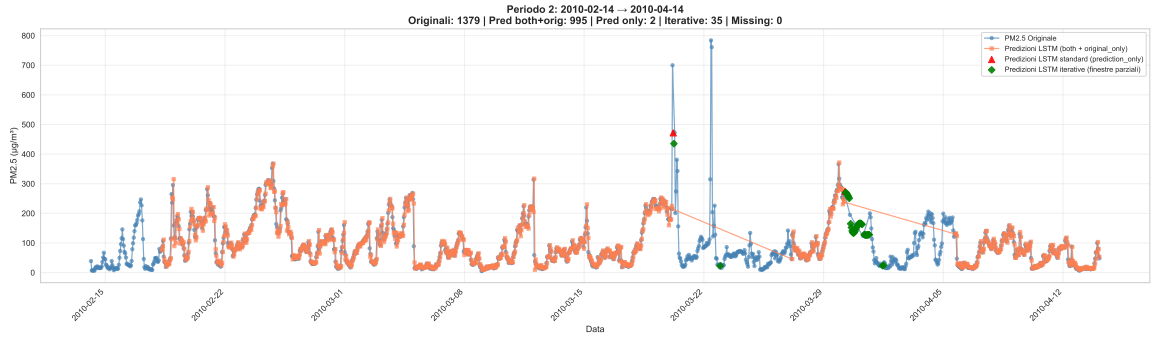
2.5 Other models evaluation

The LSTM model is the most indicated for this kind of task, but as a cross check I trained two other models to verify this assumption. Other possibilities include a vanilla Recurrent Neural Network (RNN) and a Gated Recurrent Unit (GRU), and I compared them to the LSTM model.

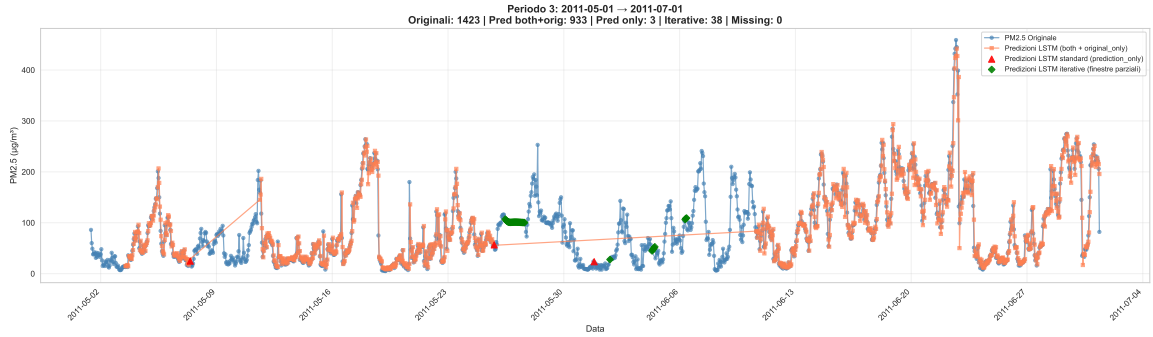
The procedure is the same as the one described in the main section, including the same data preparation and hyper parameter optimization. Since this is an exploratory analysis, I reduced the number of epochs and the number of trials in the hyperparameter optimization step.



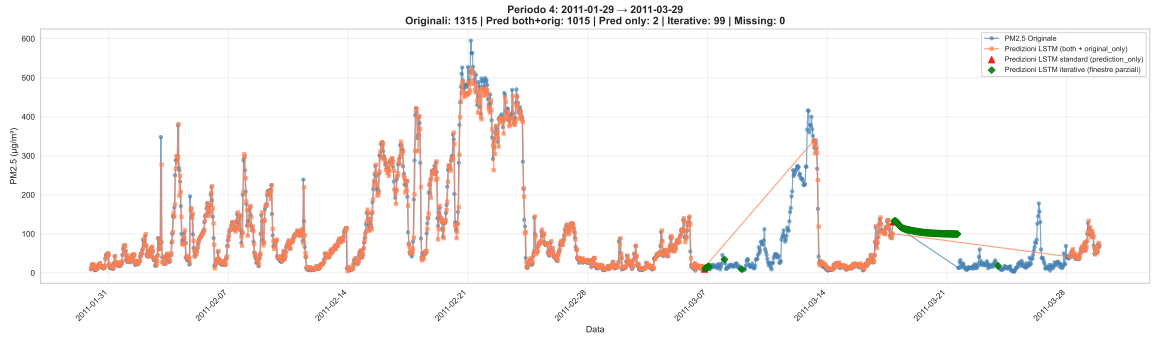
(a) Period 1



(b) Period 2



(c) Period 3



(d) Period 4

Figure 11: 4 randomly selected 2-month periods showing the detailed predictions of the LSTM model for filling missing $PM_{2.5}$ values. The plots illustrate the model's ability to accurately predict missing values using both known and previously predicted data points as input.

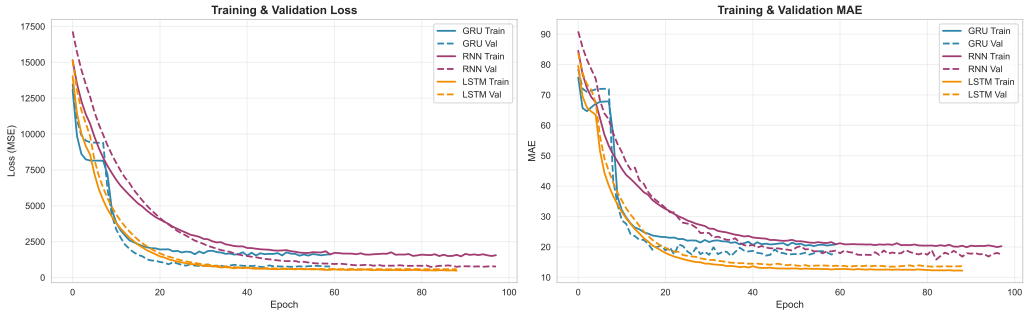


Figure 12: Training history comparison between the three models

Table 2: Test metrics to evaluate the performances across the models

Model	MSE	RMSE	MAE	R^2
GRU	670	25.9	16.1	0.910
RNN	656	25.6	14.1	0.912
LSTM	565	23.8	12.4	0.924

The best hyperparameters found for each model are summarized in Table 1.

In Figure 12 is is shown the training history of three models. The training for all of them stopped early since the condition was met. On the left, the loss (MSE) is depicted, while on the right one can see the trend of the MAE.

To evaluate the performances, the test dataset was used to calculate metrics such as MSE, MAE and R^2. The results are shown in Table 2. The LSTM model has the better parameters, in terms of MAE, RMSE and R^2. As expected, the LSTM model is the one to be preferred among the others.

3 Secondary task: classification

This dataset can also be used to perform a classification task, where the goal is to classify the air quality into different categories based on the $PM_{2.5}$ concentration levels. The air quality categories (AQI) are defined as follows, according to the US EPA standards[3]:

- Good ($0 < PM_{2.5} < 9\ \mu g/m^3$)
- Moderate ($9.1 < PM_{2.5} < 35.4\ \mu g/m^3$)
- Unhealthy for Sensitive Groups ($35.5 < PM_{2.5} < 55.4\ \mu g/m^3$)
- Unhealthy ($55.5 < PM_{2.5} < 125.4\ \mu g/m^3$)
- Very Unhealthy ($125.5 < PM_{2.5} < 225.4\ \mu g/m^3$)
- Hazardous ($225.5 < PM_{2.5}$)

Figure 13 shows the distribution of $PM_{2.5}$ classes in the dataset. In the pie chart, one can see that the most frequent category is 'Unhealthy' (31%), followed by 'Moderate' (25%) and 'Very Unhealthy' (18%). The least frequent categories are 'Hazardous' (10%) and 'Good' (4%). The dataset is quite imbalanced, with certain categories being much more prevalent than others.

It is also interesting to look at the dataset from a temporal perspective, to see how the air quality categories are distributed over time. Figure 14 shows the time series of $PM_{2.5}$ values integrated over a month, colored according to the air quality categories distribution in that month. The plot reveals seasonal patterns in air quality, with worse conditions typically observed during the winter months

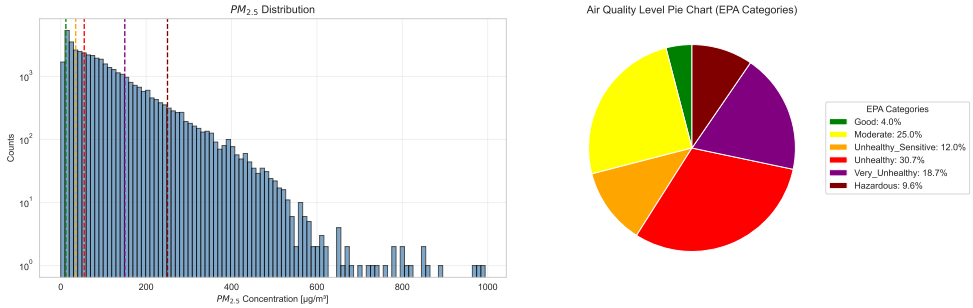


Figure 13: Distribution of $PM_{2.5}$ classes in the Beijing $PM_{2.5}$ dataset. On the left, the histogram shows the frequency of records as in Figure 3a, with vertical lines indicating the boundaries between different air quality categories. On the right, the pie chart displays the amount of records in each air quality category, highlighting the predominance of 'Unhealthy' (31%) and 'Moderate' (25%).

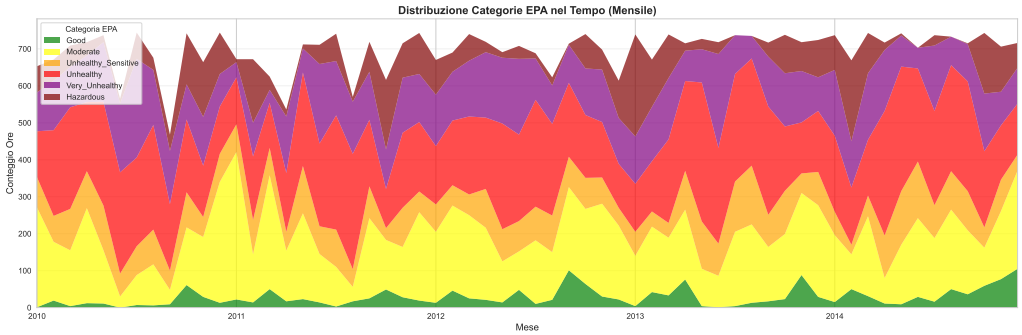


Figure 14: Time series of $PM_{2.5}$ values integrated over a month, colored according to the air quality categories distribution in that month.

(November to February) and better conditions during the summer months (June to August). This seasonal variation is likely influenced by factors such as heating practices in winter and meteorological conditions that affect pollutant dispersion.

Furthermore, towards the end of the dataset (2014), there is a noticeable improvement in air quality, with a higher frequency of 'Good' and 'Moderate' categories. This improvement could be attributed to the implementation of stricter air pollution control measures by the Beijing government starting from 2013 and showng the first results the following year.

3.1 Feature engineering and correlation

Since the feature $PM_{2.5}$ is used to define the target classes, the other meteorological features can be used to predict the air quality category. Here, features engineering and correlation studies are conducted, in order to better characterize the classifier for the AQI.

In the original paper describing the dataset [1], it was explained that the wind speed has a significant impact on the $PM_{2.5}$ concentration levels, as well as the direction of the wind. In the opionion of the authors, a new feature called `CWP` (Combined Wind Parameter) which combines the `cbwd` and `Iws` variables could better capture the influence of wind on air pollution. The `CWP` feature is created by accumulating the wind speed in a given direction, until this direction changes.

To verify the correlation between `CWP` and `pm2.5`, and also to check for other possible correlations between the features, I computed the correlation matrix of the dataset, shown in Figure 15. The correlation matrix reveals several relationships between the features:

- **Iws and CWP pretty much identical:** It is evident that the `CWP` feature is almost perfectly correlated with the `Iws` variable, and that comparing the correlation between `CWP` and the other

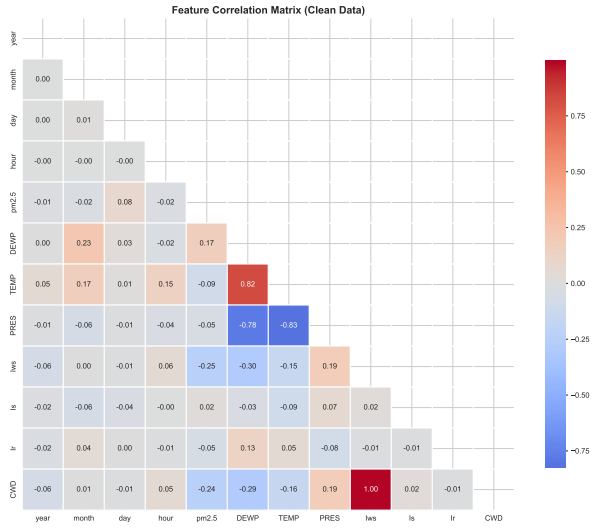


Figure 15: Correlation

variables with the correlation of `Iws` to other variables, these result identical. So the `CWP` does not provide any additional information compared to using `Iws` alone, at least in terms of linear correlation.

- **Weak negative correlation between Iws and pm2.5:** This indicates that higher wind speeds are slightly associated with lower $PM_{2.5}$ concentrations, likely due to the dispersal effect of wind on air pollutants.
- **No significant correlation between pm2.5 and other features:** except for the weak negative correlation with Iws, there are no strong correlations between $PM_{2.5}$ and the other meteorological variables. This suggests that $PM_{2.5}$ levels are influenced by a complex interplay of factors, and not solely by the individual meteorological variables measured in this dataset, at least not in a linear way.
- **Strong positive correlation between TEMP and DEWP:** This indicates that as the temperature increases, the dew point also tends to increase, suggesting a relationship between these two meteorological variables. This is expected, as warmer air can hold more moisture, leading to higher dew points.
- **Strong negative correlation between PRES and TEMP/DEWP:** This suggests that higher temperatures and dew points are associated with lower atmospheric pressure, which is consistent with meteorological principles.
- **Weak negative correlation between Iws and DEWP/TEMP:** This suggests that higher wind speeds are slightly associated with lower temperatures and dew points, which could be due to the cooling effect of wind.

Since there are some correlated features (e.g., TEMP and DEWP, PRES and TEMP/DEWP), it could be useful to perform a dimensionality reduction technique, such as Principal Component Analysis (PCA), to reduce the number of features while retaining most of the variance in the data. PCA was performed on the standardized meteorological features (DEWP, TEMP, PRES, Iws, Is, Ir, CWD), excluding the temporal variables and the target variable pm2.5. The results of the PCA are shown in Figures 16 and 17, and as well in Table 3. Figure 16 shows the explained variance ratio for each principal component. The first principal component (PC1) explains roughly 40% of the variance of the data, while the second principal component (PC2) adds another 25% of explained variance.

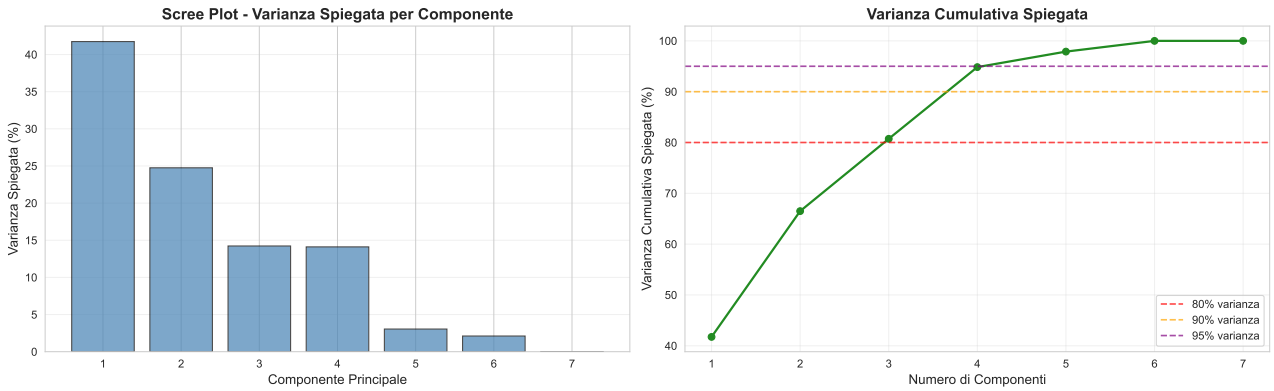


Figure 16: PCA

Adding the third (PC3) and fourth (PC4) principal components brings the total explained variance to about 95%.

This means that we can reduce the dimensionality of the dataset from 7 meteorological features to 4 principal components, while still retaining most of the information in the data. Table 3 shows the loadings of the original features on the first 4 principal components. From this table, we can see how each original feature contributes to each principal component. For example, PC1 is heavily positively influenced by TEMP and DEWP, and negatively influenced by PRES and Iws (CWD). This could suggest that this takes into account the general "warmness" of the air, with higher temperatures and dew points associated with lower pressures and wind speeds.

Table 3: PCA Loadings

	DEWP	TEMP	PRES	Iws	Is	Ir	CWD	Variance (%)
PC1	0.52	0.49	-0.49	-0.34	-0.06	0.07	-0.34	41.74
PC2	0.21	0.33	-0.30	0.61	-0.04	0.07	0.61	24.75
PC3	0.07	-0.04	0.00	-0.00	0.85	0.52	-0.00	14.23
PC4	-0.04	-0.09	0.07	-0.02	-0.52	0.84	-0.02	14.11

We can also look at the correlation between the principal components and the original features, as shown in Figure 17, to see if any of the combined features have a stronger correlation with the target variable pm2.5. From this plot, we can see that none of the principal components have a strong correlation with pm2.5, similar to the original features. The highest (negative) correlation is seen with PC6, which explains only a small fraction of the variance in the data. The scatter plots of pm2.5 against PC6 is depicted on the right side of Figure 17, showing no particular evident trend or pattern.

This suggests that while PCA can help reduce the dimensionality of the dataset, it does not necessarily lead to features that are more predictive of $PM_{2.5}$ levels in a linear sense. Furthermore, the space was reduced from 6 (since the seventh was artificially created and practically identical to another feature) to 4 dimensions, which is not a significant reduction.

It was therefore chosen to use the original meteorological feature, with the exception of `CWD`, which is practically identical to `Iws`, but takes also into account the wind direction, which otherwise should have been one-hot encoded.

Since the past values may also have an influence on the current air quality, I also created lag features for each meteorological variable. Specifically, for each meteorological variable, I created lag features from 1 to 12 hours (i.e., the values of the variable at the previous 1 to 12 time steps). This allows the model to capture temporal dependencies and trends in the meteorological data that may affect $PM_{2.5}$ levels and hence the air quality category.

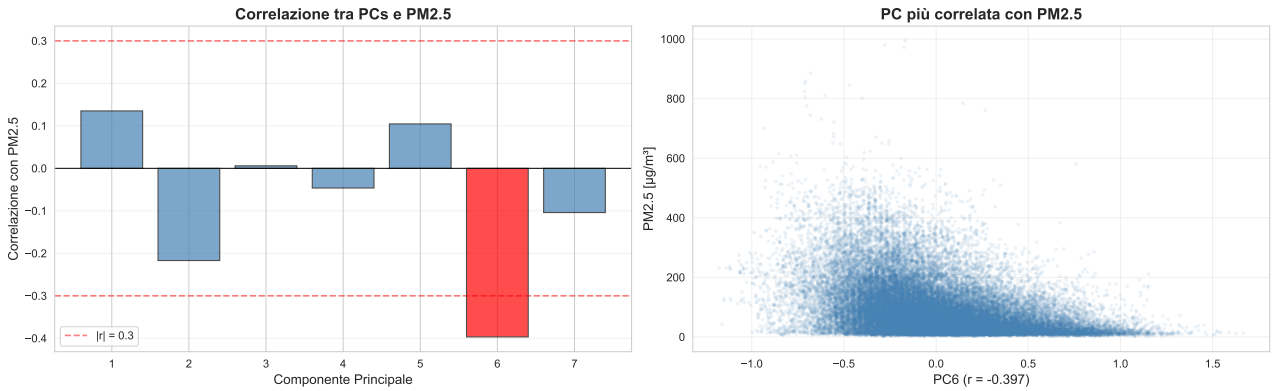


Figure 17: PCA Correlation

The seasonality, the period of day/week may also have an impact to the class. Therefore, I decided to create 9 other boolean variables such as:

- `is_night`: takes into account whether the hour is between 0 and 6.
- `is_morning`: takes into account whether the hour is between 6 and 12.
- `is_afternoon`: takes into account whether the hour is between 12 and 18.
- `is_evening`: takes into account whether the hour is between 18 and 24.
- `is_weekend`: takes into account whether the day is saturday or sunday.
- `is_winter`: takes into account whether the month is december, january or february.
- `is_spring`: takes into account whether the month is march, april or may.
- `is_summer`: takes into account whether the month is june, july or august.
- `is_autumn`: takes into account whether the month is september, october or november.

In order to not generalize too much, I kept the initial information about hour, day, month and year as numerical variables, since there may also be some trends over the years (for example, the air quality improved in the last years of the dataset, due to the measures taken by the Beijing government to reduce pollution).

In the end, the final feature set for the classification task consists of the original meteorological features (DEWP, TEMP, PRES, Iws, Is, Ir, CWD), their corresponding lag features from 1 to 12 hours, and the temporal variables, plus the 9 boolean variables representing different times of day, weekend, and seasons, resulting in a total of 101 features (11 original features + 7 features * 12 lags + 9 boolean variables).

3.2 Modeling

The model is implemented using a Deep Neural Network (DNN) architecture with multiple hidden layers with the PyTorch library. The network consists of an input layer, several hidden layers, and an output layer to predict the air quality category.

For this classification task, I employed the same hyperparameter optimization strategy using `Optuna` as described in the regression task. The hyperparameters that were optimized include:

- Number of hidden layers (2 to 5)
- Number of units in each hidden layer (from 32 to 1024, step 32)

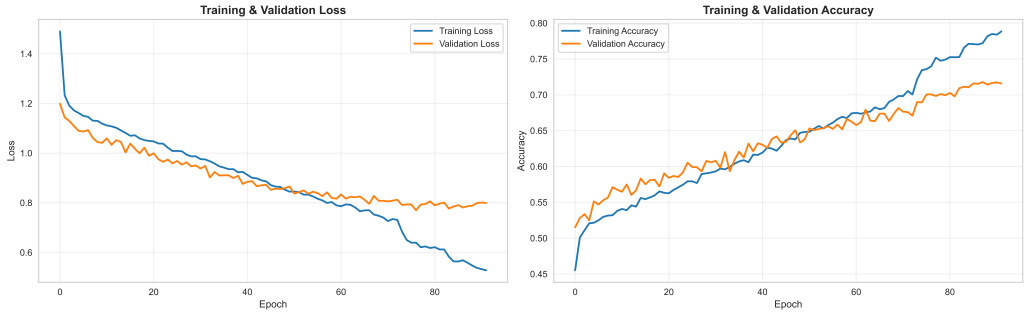


Figure 18: DNN Training history

- Dropout rate (0.0 to 0.5)
- Learning rate (1e-4 to 1e-1)
- Batch size (64, 128, 256)
- Activation function: [relu, elu, leaky relu, selu]

The best hyperparameter configuration found is the following:

- 5 hidden layers
- Hidden sizes: [992, 800, 832, 736, 320]
- Dropout rates: [0.3, 0.2, 0.2, 0.5, 0.5]
- Learning rate: 2.49e-02
- Batch size: 256
- Activation function: relu

These hyperparameters were used to train the final DNN model on the training set, with a validation split of 20% of the training data. The model was trained using the Adam optimizer and categorical cross-entropy as the loss function, with early stopping and learning rate reduction callbacks similar to those used in the regression task. The final layer uses a linear activation function to output the class probabilities for each air quality category. The model was trained for a maximum of 250 epochs, but the training stopped after 94 epochs due to early stopping.

The training history of the DNN model is shown in Figure 18, which displays the training and validation accuracy and loss over epochs on the left, and the accuracy on the right.

3.3 Results

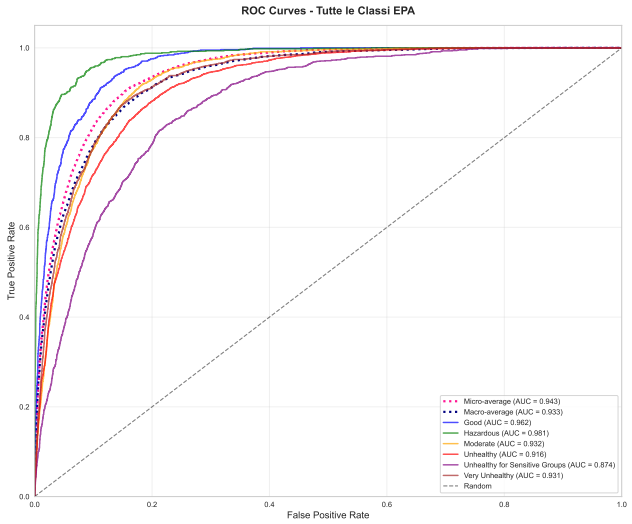
The performance of the DNN model on the test set is evaluated using various metrics, including accuracy, precision, recall, and F1-score.

These metrics are displayed in the classification report shown in Table 4. The overall accuracy of the model on the test set is 70.18%, indicating that the model is able to correctly classify the air quality category for a significant portion of the test samples.

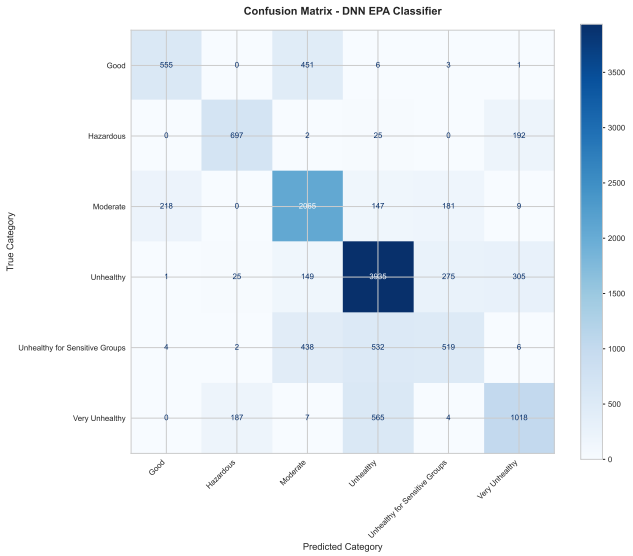
In Figure 19, the ROC curves (left) and confusion matrix (right) for the DNN model on the test set are presented. The ROC curves allows to have a bettter glance at the true and false positive rates of all the classes. Furhtermore, the micro and macro average are also displayed. The area under the ROC curve (AUC) for each class indicates the model's effectiveness in correctly classifying samples from that class, given a numerical score. From these curves, one can see that the class "Unhealthy for sensitive groups" is the one that has the worst performance, while "Hazardous" gives the best

Table 4: Classification Report

	precision	recall	f1-score	support
Good	0.7134	0.5463	0.6187	1016
Moderate	0.6636	0.7882	0.7205	2620
Unhealthy	0.7553	0.8390	0.7949	4690
Unhealthy for Sensitive Groups	0.5285	0.3458	0.4180	1501
Very Unhealthy	0.6649	0.5716	0.6147	1781
Hazardous	0.7651	0.7609	0.7630	916
accuracy			0.7018	12524
macro avg	0.6818	0.6420	0.6550	12524
weighted avg	0.6934	0.7018	0.6919	12524



(a) ROC Curves of each class, including micro and macro average



(b) Confusion Matrix

Figure 19: DNN Model Evaluation on Test Set

results. This quite surprising, given that "Hazardous" is one of the classes with the lowest support in the dataset.

The misclassifications is better understood in the confusion matrix on the right. The confusion matrix provides a detailed view of the model's performance across different classes, showing how many samples were correctly classified and where misclassifications occurred. From this confusion matrix, one can see that the category "Good" is often misclassified as "Moderate", and "Unhealthy for Sensitive Groups" is frequently confused with "Moderate" and "Unhealthy", almost in equal parts.

Overall, the DNN model demonstrates good performance in classifying air quality categories based on meteorological features, despite the challenges posed by class imbalance and overlapping characteristics between certain categories.

4 Conclusions

The tasks of this project were to predict $PM_{2.5}$ concentration values and to perform classification into Air Quality Index, using deep models for the both supervised tasks.

First an exploration of the dataset is conducted to assess the data quality and characterize the features. The dataset is also engineered to handle missing values and to give the correct format to the

models used, either for prediction or classification.

For the prediction, three different models were tested (RNN, GRU, LSTM), with different hyperparameters configuration. The best model found was a simple LSTM with a single layer. This network has good prediction propoerties, but has difficulties to predict higher concentration values.

Regarding classifications, the classes are created using intervals of the $PM_{2.5}$, as is sone in the AQI index of the EPA. The features were engineered to account for the seasonal variations and to account for the previous values of the meteorological features. Then the data was treated as a tabular dataset and a DNN architecture with 5 layers was implemented. The results are not optimal, with an overall accuracy of 70%. The model struggles the most with the category "Unhealthy for Sensitive Groups" and "Good". To increase the accuracy, in a following studies we could think of better balancing the classes

Furthermore, the model used for classification was not really suited for this kind of task. I tried to engineer the features in order to reconduct it to a task that was simple (tabular dataset), but other models may have performed better. An example is the InceptionTime architecture, but since it is complicated and this kind of tasks was not treated in class, I decided to not go for this way.

Overall, the tasks are accomplished with quite good results.

5 References

[1] Xuan Liang, Tao Zou, Bin Guo, Shuo Li, Haozhe Zhang, Shuyi Zhang, Hui Huang, and Song Xi Chen. Assessing beijing's pm2.5 pollution: severity, weather impact, apec and winter heating. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471, 2015. URL `https://api.semanticscholar.org/CorpusID:130615236`.

[2] Wikipedia contributors. 2010 china drought and dust storms — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=2010_China_drought_and_dust_storms&oldid=1320461493`, 2025. [Online; accessed 9-December-2025].

[3] Wikipedia contributors. Air quality index — Wikipedia, the free encyclopedia, 2025. URL `https://en.wikipedia.org/w/index.php?title=Air_quality_index&oldid=1326748082`. [Online; accessed 11-December-2025].

[4] Wikipedia contributors. Chinese new year — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Chinese_New_Year&oldid=1326818228`, 2025. [Online; accessed 12-December-2025].