# Advanced Programming

**Final Project**

**Tyo Ignat, Agybaev Aidyn, Alisher Galymzhan**

**IT-2206**

**Weather forecast and house price prediction**

## PART 1

## Weather forecast & weather image classification

### Introduction:

**Problem**: We chose to work with weather and weather datasets because this topic provides many opportunities for implementing machine learning. When forecasting weather, many factors are taken into account, not just temperature and wind speed. For example: precipitation, snow depth, humidity, cloudiness, wind direction and others. All these factors influence the formation and perception of weather in general. Using this data, a machine learning model can be trained to predict the weather. By changing certain indicators, the accuracy and indications of the forecast will differ. Thus, it is possible to identify which specific factors are most important in weather forecasting.

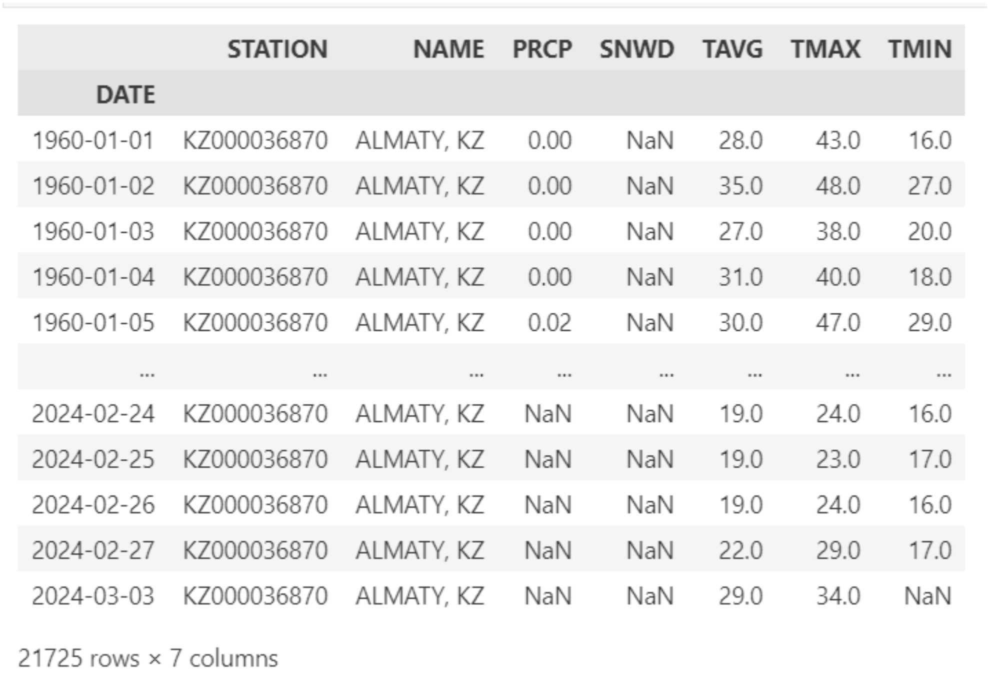
**Another Solutions:** Currently, there are many services dedicated to weather forecasts. Some of them also provide data for free use, such as Open Weather API. The main functionality of the OpenWeather API is to provide access to weather data and forecasts worldwide. It offers current weather conditions, forecasts, historical data, and weather maps, along with various weather-related data such as temperature, humidity, wind speed, and more. Developers can integrate this API into their applications to retrieve accurate and up-to-date weather information for specific locations.

https://openweathermap.org/api

**Current work**: Our work with weather consists of two parts: weather prediction (maximum and minimum temperatures) and classification of images by weather. In our project, we used three models to predict weather for three different cities: Astana, Almaty and Karaganda. These are identical models, trained on a weather datasets from 1960 to 2024 (for different cities) from the NOAA website (https://www.ncei.noaa.gov/cdo-web/search), based on Ridge. To implement image classification, Tensorflow keras was used. These models were then implemented on the website using Node.js. The site also has a weather search for a specific city using the Open Weather API.

### Data and Methods:

For the weather prediction model, 3 datasets were used with weather forecasts for different cities: Astana, Almaty and Karaganda for the period from 1960 to 2014. All 3 weather forecasting models for the 3 cities were implemented in the same way.

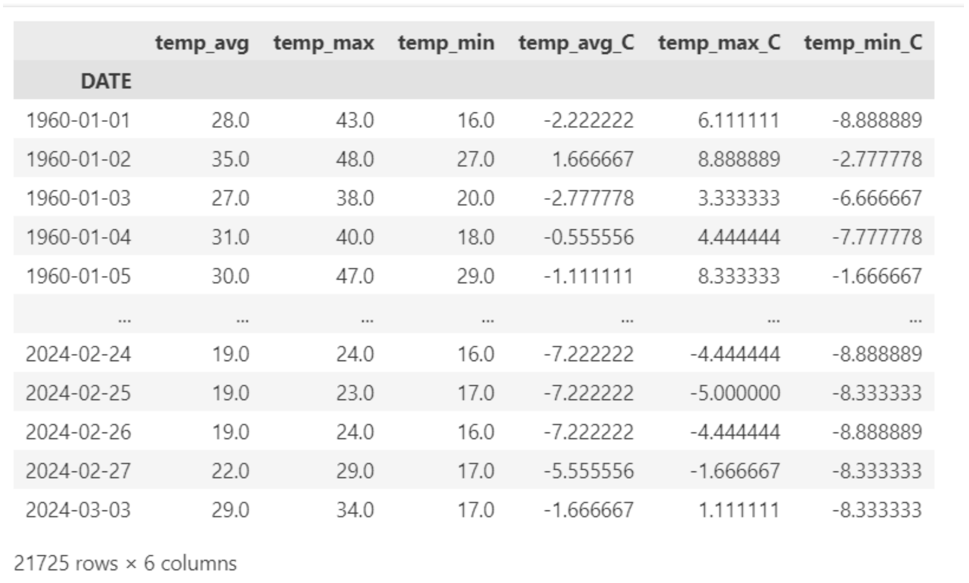


| | STATION | NAME | PRCP | SNWD | TAVG | TMAX | TMIN |
|---|---|---|---|---|---|---|---|
| DATE | | | | | | | |
| 1960-01-01 | KZ000036870 | ALMATY, KZ | 0.00 | NaN | 28.0 | 43.0 | 16.0 |
| 1960-01-02 | KZ000036870 | ALMATY, KZ | 0.00 | NaN | 35.0 | 48.0 | 27.0 |
| 1960-01-03 | KZ000036870 | ALMATY, KZ | 0.00 | NaN | 27.0 | 38.0 | 20.0 |
| 1960-01-04 | KZ000036870 | ALMATY, KZ | 0.00 | NaN | 31.0 | 40.0 | 18.0 |
| 1960-01-05 | KZ000036870 | ALMATY, KZ | 0.02 | NaN | 30.0 | 47.0 | 29.0 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 2024-02-24 | KZ000036870 | ALMATY, KZ | NaN | NaN | 19.0 | 24.0 | 16.0 |
| 2024-02-25 | KZ000036870 | ALMATY, KZ | NaN | NaN | 19.0 | 23.0 | 17.0 |
| 2024-02-26 | KZ000036870 | ALMATY, KZ | NaN | NaN | 19.0 | 24.0 | 16.0 |
| 2024-02-27 | KZ000036870 | ALMATY, KZ | NaN | NaN | 22.0 | 29.0 | 17.0 |
| 2024-03-03 | KZ000036870 | ALMATY, KZ | NaN | NaN | 29.0 | 34.0 | NaN |

21725 rows × 7 columns

First, we analyzed the existing date and selected only the one suitable for training our model. We discarded columns of information that contained a lot of NULL values. If there were few NULL values in the column, we replaced it with the value from the previous row (based on the fact that the weather did not change much in one day). Since the temperature in the dataset was initially given in Faringheit, we added new columns where we converted this data to Celsius.

The dataset, ready for training, looked like this:



| | temp_avg | temp_max | temp_min | temp_avg_C | temp_max_C | temp_min_C |
|---|---|---|---|---|---|---|
| DATE | | | | | | |
| 1960-01-01 | 28.0 | 43.0 | 16.0 | -2.222222 | 6.111111 | -8.888889 |
| 1960-01-02 | 35.0 | 48.0 | 27.0 | 1.666667 | 8.888889 | -2.777778 |
| 1960-01-03 | 27.0 | 38.0 | 20.0 | -2.777778 | 3.333333 | -6.666667 |
| 1960-01-04 | 31.0 | 40.0 | 18.0 | -0.555556 | 4.444444 | -7.777778 |
| 1960-01-05 | 30.0 | 47.0 | 29.0 | -1.111111 | 8.333333 | -1.666667 |
| ... | ... | ... | ... | ... | ... | ... |
| 2024-02-24 | 19.0 | 24.0 | 16.0 | -7.222222 | -4.444444 | -8.888889 |
| 2024-02-25 | 19.0 | 23.0 | 17.0 | -7.222222 | -5.000000 | -8.333333 |
| 2024-02-26 | 19.0 | 24.0 | 16.0 | -7.222222 | -4.444444 | -8.888889 |
| 2024-02-27 | 22.0 | 29.0 | 17.0 | -5.555556 | -1.666667 | -8.333333 |
| 2024-03-03 | 29.0 | 34.0 | 17.0 | -1.666667 | 1.111111 | -8.333333 |

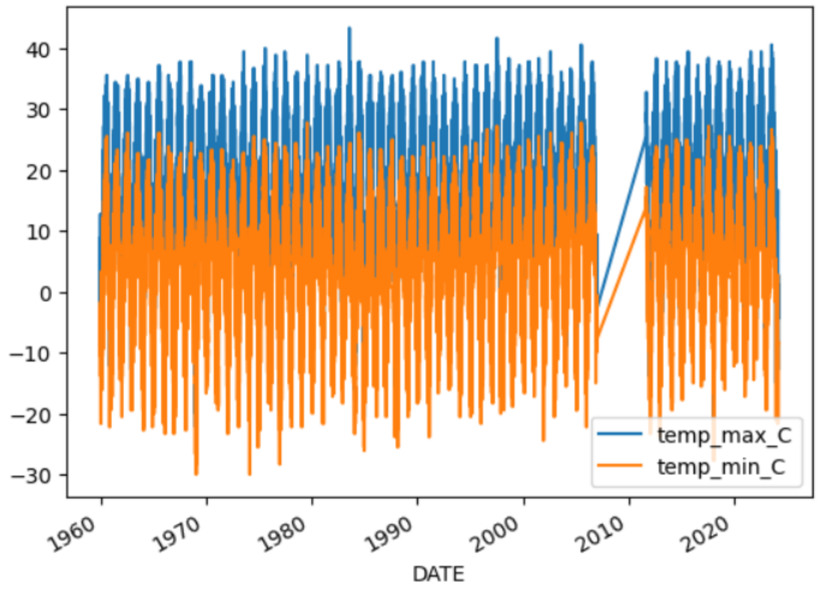
21725 rows × 6 columns

Visualisation of data:

```
core_weather[["temp_max_C", "temp_min_C"]].plot()
```

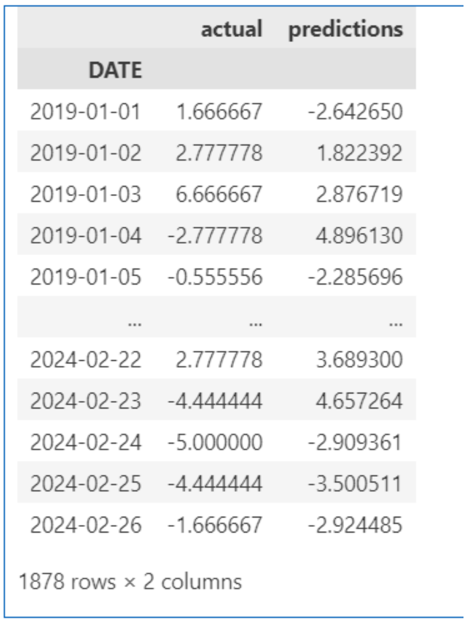
[14]

```
<Axes: xlabel='DATE'>
```



Then, we set a target for each maximum temperature equal to the maximum temperature of the next day, and started to train the model using sklearn linear model Ridge. As training progressed, we added more and more data to train the model using future engineering. We added new data on which the prediction was made: "temp_avg_C", "temp_max_C", "temp_min_C", "month_day_max", "max_min", "month_max", "day_of_year_avg", "monthly_avg", "temp_range", "temp_avg_diff". All this data was calculated from initially only three columns: maximum, minimum and average temperature for the day.

Results: At the end of the training, our model gave us the result with weather prediction for dates from 2019-01-01 to 2024-02-26.

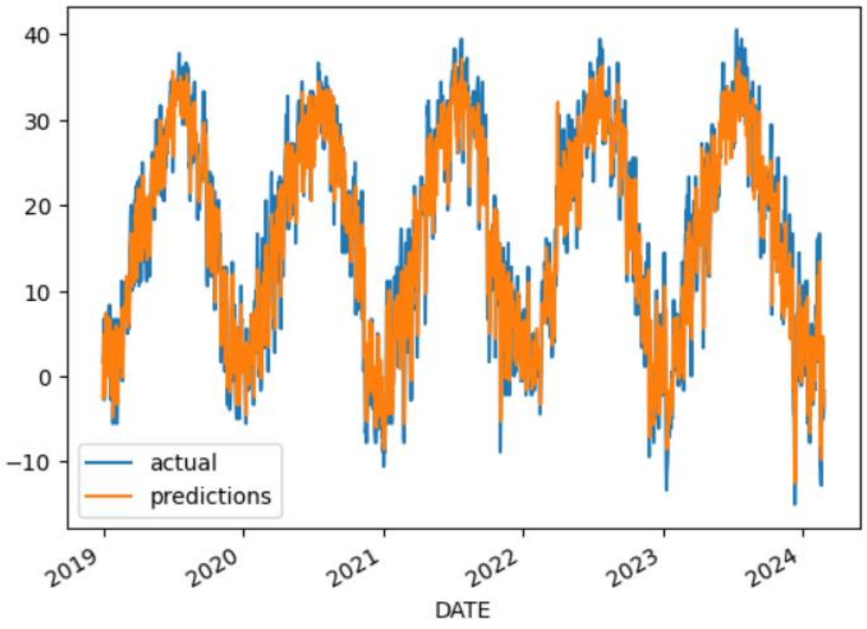


| | actual | predictions |
|---|---|---|
| DATE | | |
| 2019-01-01 | 1.666667 | -2.642650 |
| 2019-01-02 | 2.777778 | 1.822392 |
| 2019-01-03 | 6.666667 | 2.876719 |
| 2019-01-04 | -2.777778 | 4.896130 |
| 2019-01-05 | -0.555556 | -2.285696 |
| ... | ... | ... |
| 2024-02-22 | 2.777778 | 3.689300 |
| 2024-02-23 | -4.444444 | 4.657264 |
| 2024-02-24 | -5.000000 | -2.909361 |
| 2024-02-25 | -4.444444 | -3.500511 |
| 2024-02-26 | -1.666667 | -2.924485 |

1878 rows × 2 columns

```
combined.plot()
```

```
<Axes: xlabel='DATE'>
```



Following the same path, we also trained the model to predict the minimum temperature.

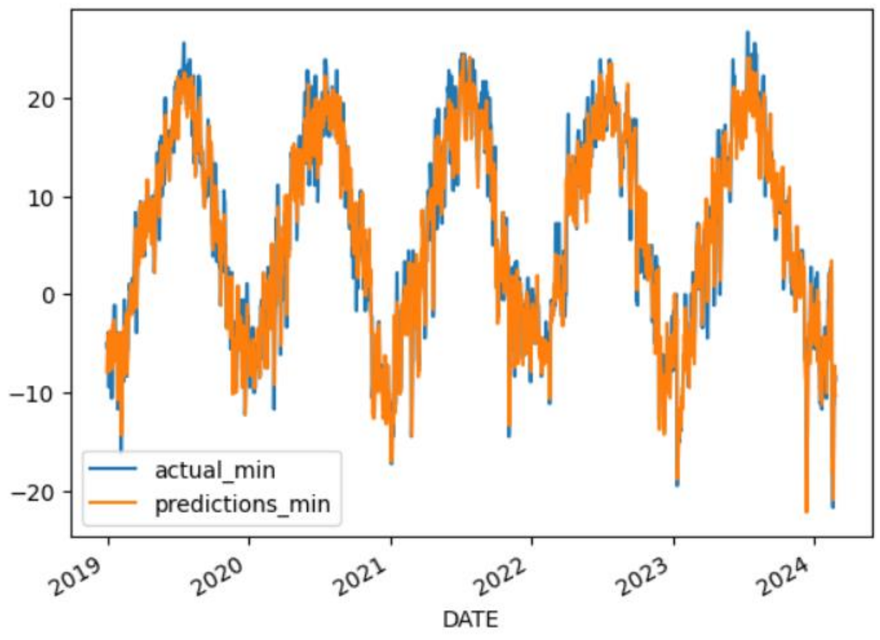
```
combined_min
```

| | actual_min | predictions_min |
|---|---|---|
| **DATE** | | |
| 2019-01-01 | -5.000000 | -7.894423 |
| 2019-01-02 | -5.555556 | -5.909851 |
| 2019-01-03 | -3.888889 | -5.629466 |
| 2019-01-04 | -3.888889 | -4.055330 |
| 2019-01-05 | -3.888889 | -7.266028 |
| ... | ... | ... |
| 2024-02-22 | -11.111111 | -7.961865 |
| 2024-02-23 | -8.888889 | -7.246232 |
| 2024-02-24 | -8.333333 | -10.252919 |
| 2024-02-25 | -8.888889 | -10.179321 |
| 2024-02-26 | -8.333333 | -10.269832 |

1878 rows × 2 columns

```
combined_min.plot()
```

```
<Axes: xlabel='DATE'>
```



Predictios for a date not in the original dataset:

```
Predicted temp_min_C for 2024-02-27: [-10.2711429]
```

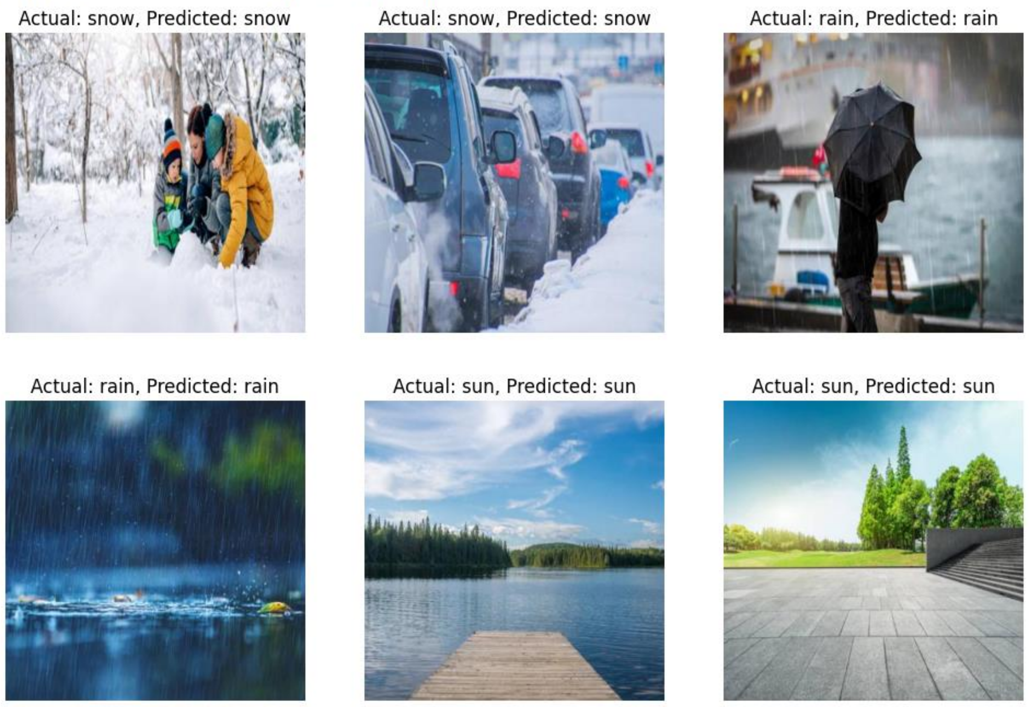
```
Predicted temp_max_C for 2024-02-27: [-2.92629114]
```

**This was a description of weather forecasting for one city (Almaty). Weather forecasting models for other cities were created in the same way.**

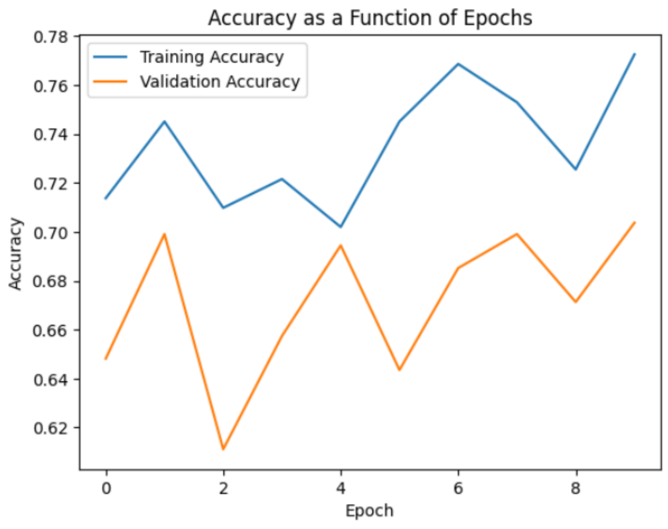
## Image Classification:

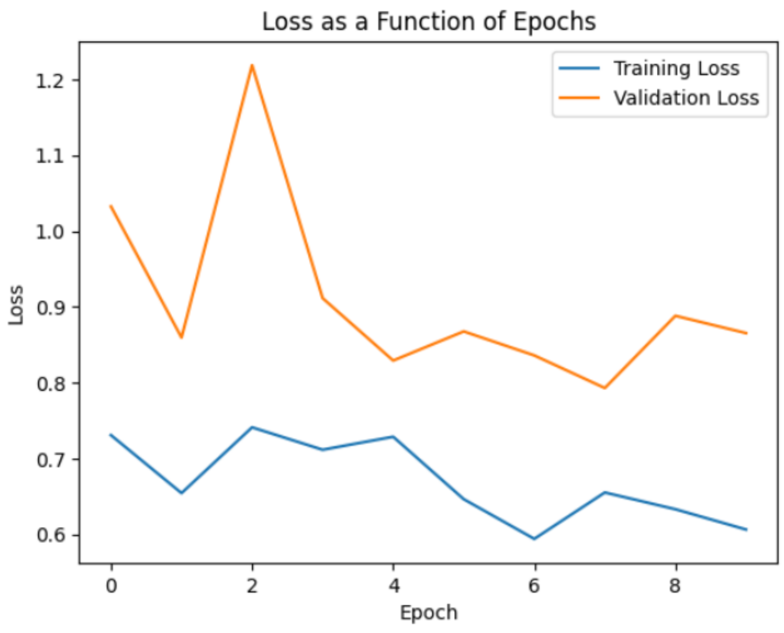
To train the model for image classification, we used TensorFlow. We collected our dataset manually from stock images on the Internet. The total number of images is approximately 400; approximately 50 images were used for training and validation of a certain class of images. The model classifies images into four classes: fog, sun, rain and snow. Towards the end of the training, we added date augmentation to our model. After training the model, we checked the plot accuracy and loss as functions of epochs.

```
1/1 [==============================] - 0s 107ms/step
```

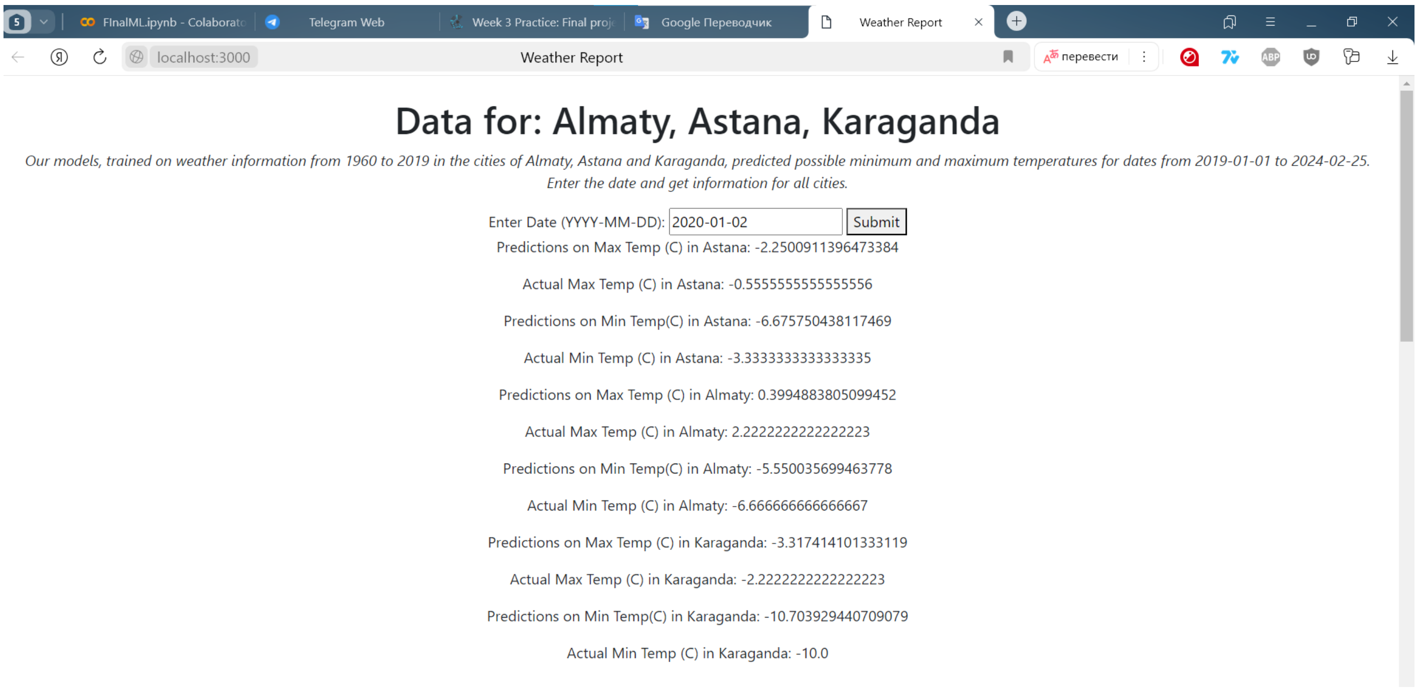

Actual: snow, Predicted: snow
Actual: snow, Predicted: snow
Actual: rain, Predicted: rain
Actual: rain, Predicted: rain
Actual: sun, Predicted: sun
Actual: sun, Predicted: sun

```
Epoch 10/10
8/8 [==============================] - 101s 13s/step - loss: 0.6064 - accuracy: 0.7725 - v
```





After training the models, we implemented them on a web page using Node.js.



## Data for: Almaty, Astana, Karaganda

*Our models, trained on weather information from 1960 to 2019 in the cities of Almaty, Astana and Karaganda, predicted possible minimum and maximum temperatures for dates from 2019-01-01 to 2024-02-25. Enter the date and get information for all cities.*

Enter Date (YYYY-MM-DD): 2020-01-02 Submit

Predictions on Max Temp (C) in Astana: -2.2500911396473384

Actual Max Temp (C) in Astana: -0.5555555555555556

Predictions on Min Temp(C) in Astana: -6.675750438117469

Actual Min Temp (C) in Astana: -3.3333333333333335

Predictions on Max Temp (C) in Almaty: 0.3994883805099452

Actual Max Temp (C) in Almaty: 2.2222222222222223

Predictions on Min Temp(C) in Almaty: -5.550035699463778

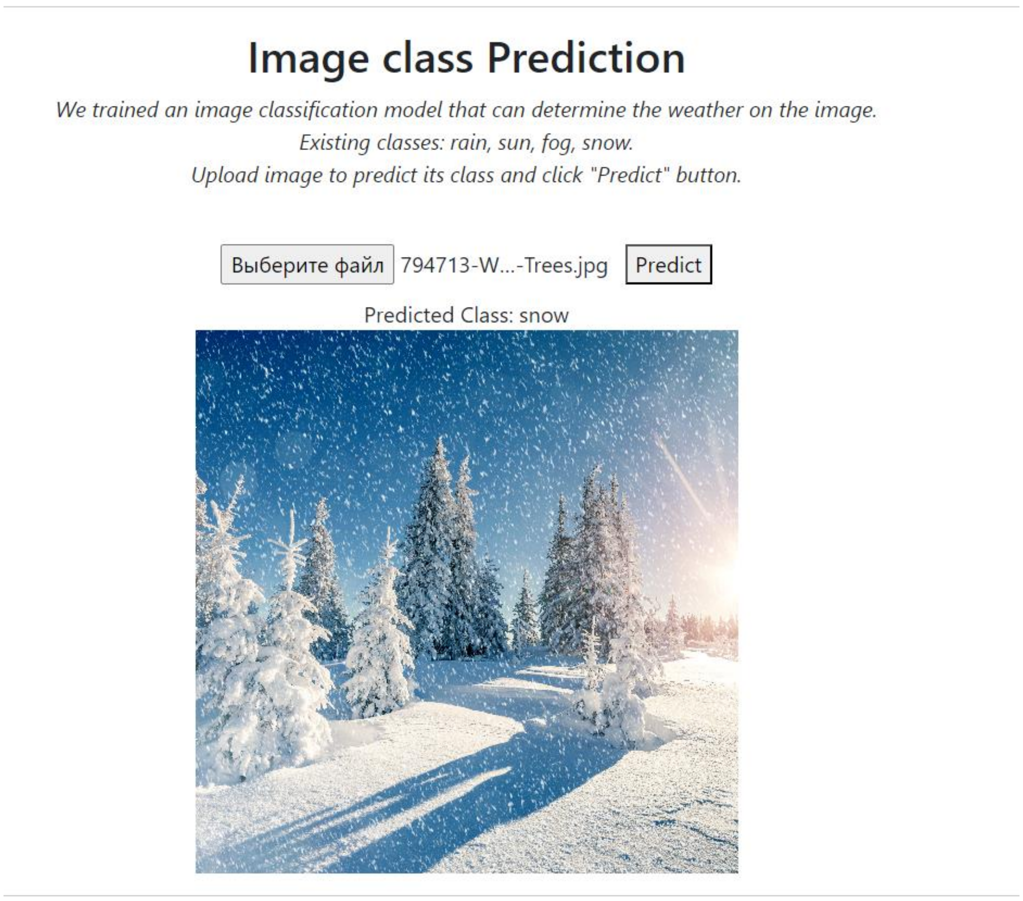
Actual Min Temp (C) in Almaty: -6.666666666666667

Predictions on Max Temp (C) in Karaganda: -3.317414101333119

Actual Max Temp (C) in Karaganda: -2.2222222222222223

Predictions on Min Temp(C) in Karaganda: -10.703929440709079

Actual Min Temp (C) in Karaganda: -10.0



## Image class Prediction

*We trained an image classification model that can determine the weather on the image.*
*Existing classes: rain, sun, fog, snow.*
*Upload image to predict its class and click "Predict" button.*

Выберите файл 794713-W...-Trees.jpg Predict

Predicted Class: snow

## Discussion:

We created models to predict temperature for specific cities and a model to classify images. Both of these models can be further improved. For example: train models to make predictions for more cities; add prediction of not only temperature, but also, for example, pressure or precipitation; add more weather classes to the image classification model; improve the accuracy of the classification model using more complex training methods. However, we believe that the prediction accuracy of our models is quite good for small models, because a difference of just a few percent is a good result for weather prediction. And when classifying images, our model only has big difficulties with fog images - the accuracy of other classes is quite high. We believe that our project can be a good basis (or at least an idea) for a full-fledged website dedicated to the weather.

# PART 2

# House price prediction

## Introduction

### Problem

Predicting house prices is a common problem in real estate and finance. It involves developing a model that can estimate the selling price of a house based on various features like, location, infrastructure, age of the house and etc.

### Existing solutions

One of the most popular solutions on this field is ‘Krisha.kz’. This is a site where an ordinary user can rent or sell his apartment, as well as find the desired apartment using certain filters. It also provides the opportunity for the program to determine the approximate cost of the apartment depending on its characteristics

**https://krisha.kz/**

### Current work

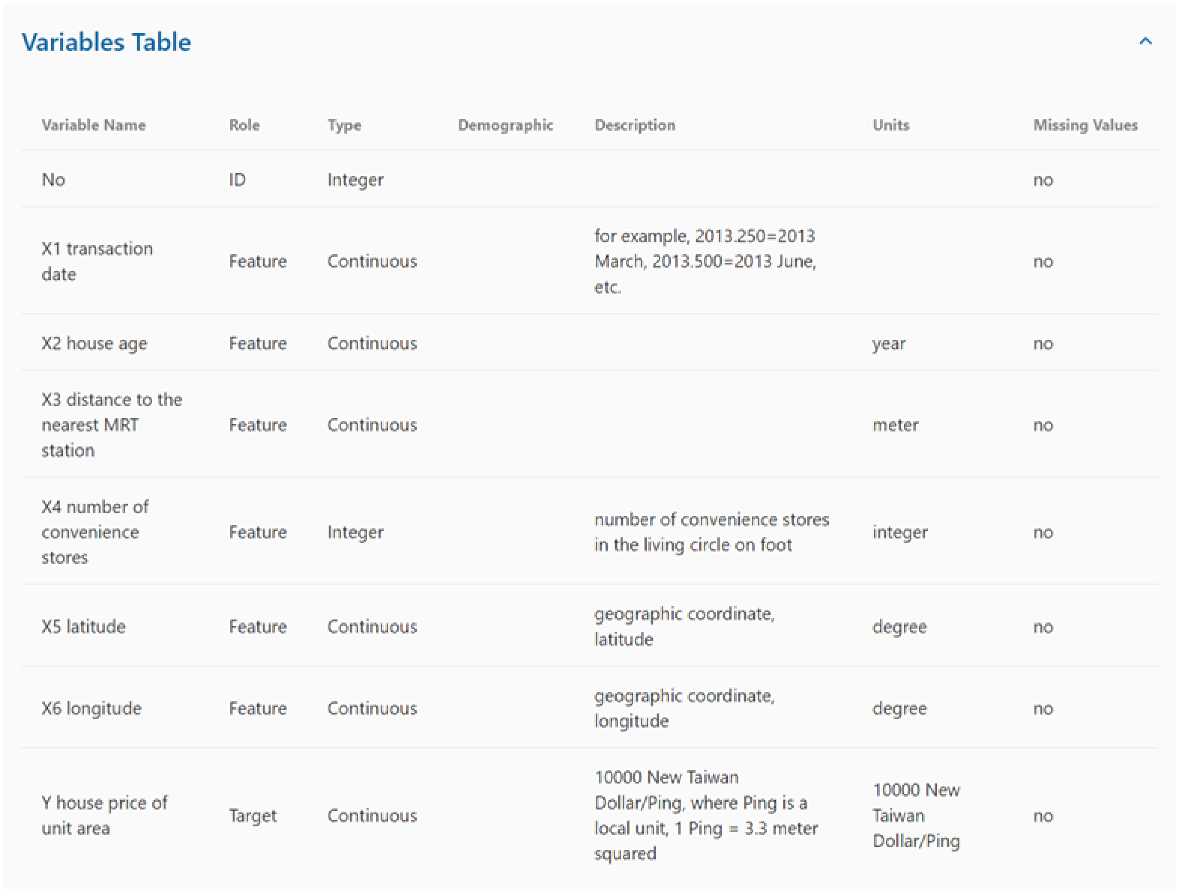
The main goal of our project is to create a deep learning model that will give the approximate cost of an apartment depending on its characteristics, and also further use this model in a web application

## Data and methods

### Information about data

This dataset was downloaded from a free resource that allows you to find data on any topic. It consists of an Excel file where all the main characteristics are shown in columns

https://archive.ics.uci.edu/dataset/477/real+estate+valuation+data+set



Variables Table

| Variable Name | Role | Type | Demographic | Description | Units | Missing Values |
|---|---|---|---|---|---|---|
| No | ID | Integer | | | | no |
| X1 transaction date | Feature | Continuous | | for example, 2013.250=2013 March, 2013.500=2013 June, etc. | | no |
| X2 house age | Feature | Continuous | | | year | no |
| X3 distance to the nearest MRT station | Feature | Continuous | | | meter | no |
| X4 number of convenience stores | Feature | Integer | | number of convenience stores in the living circle on foot | integer | no |
| X5 latitude | Feature | Continuous | | geographic coordinate, latitude | degree | no |
| X6 longitude | Feature | Continuous | | geographic coordinate, longitude | degree | no |
| Y house price of unit area | Target | Continuous | | 10000 New Taiwan Dollar/Ping, where Ping is a local unit, 1 Ping = 3.3 meter squared | 10000 New Taiwan Dollar/Ping | no |

**Description of ML/DL model**

Written code is a Python script using the TensorFlow and scikit-learn libraries to build and train a neural network for predicting house prices based on real estate features. Here's a description of the ML/DL model part code:

**Data Splitting:**

Splits the dataset into features (X) and the target variable (y).

Further splits the data into training and testing sets using the train_test_split function.

**Preprocessing Numerical Features:**

Uses StandardScaler to standardize the numerical features of the training set (X_train).

**Building the Neural Network Model:**

Creates a sequential model using Keras.

**Adds three layers to the model:**

The first layer with 64 neurons, ReLU activation function, and an input dimension corresponding to the number of features in the training set.

The second layer with 32 neurons and ReLU activation function.

The output layer with 1 neuron (since it's a regression task) and a linear activation function.

**Compiling the Model:**

Compiles the model using the Adam optimizer, mean squared error (MSE) as the loss function, and mean absolute error (MAE) as the metric to monitor during training.

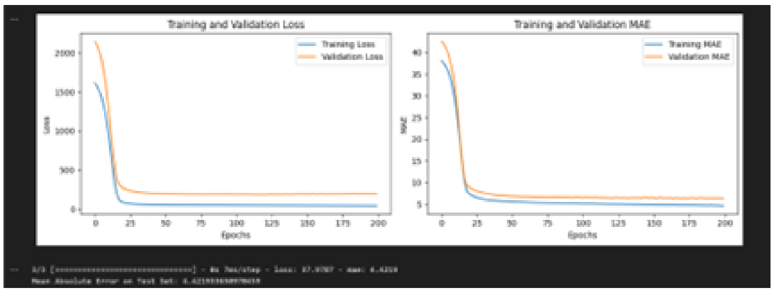
**Training the Model:**

Fits the model to the preprocessed training data with early stopping.

Uses a batch size of 32 and trains for 200 epochs.

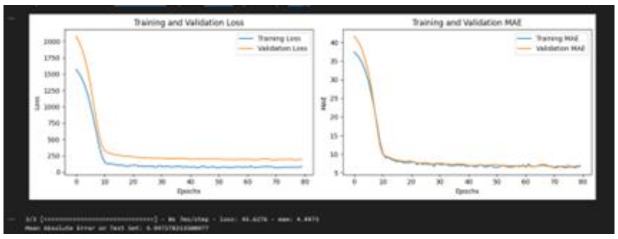
Monitors validation loss and stops training early if it doesn't improve.

Amongst many different models, model with both accuracy and efficiency was chosen



```
3/3 [==============================] - 0s 7ms/step - loss: 37.9787 - mae: 4.4219
Mean Absolute Error on Test Set: 4.421933689978039
```

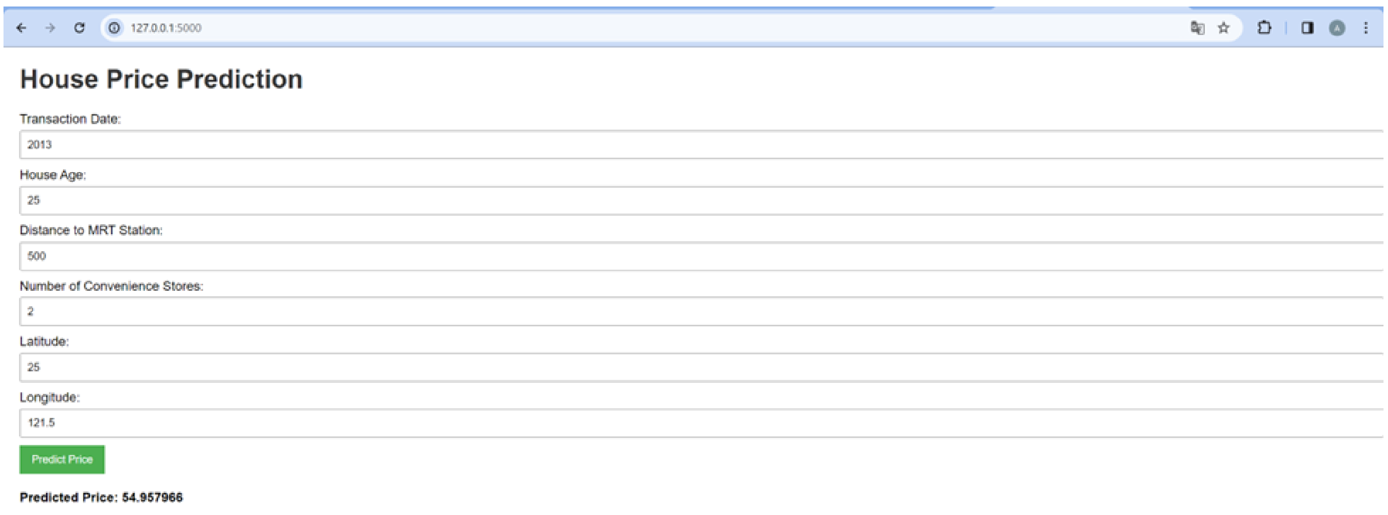
*Figure 1-Plot of the final model*



```
3/3 [==============================] - 0s 7ms/step - loss: 46.6276 - mae: 4.4971
Mean Absolute Error on Test Set: 4.497178212088077
```

*Figure 2-Plot of the one of the models*

## Results


127.0.0.1:5000

House Price Prediction

Transaction Date: 2013

House Age: 25

Distance to MRT Station: 500

Number of Convenience Stores: 2

Latitude: 25

Longitude: 121.5

**Predicted Price: 54.957966**

The result is a functional website with the ability to find out the price of your home. For this, in addition to the model itself, a website was made based on html, javascript and flask

## Discussion

This project can still be improved. For example, use a more extensive dataset to improve the accuracy of the model. Also, the model itself can be improved by adding more functionality and details. More advanced interface can be also added

## REFERENCES

- https://www.databricks.com/glossary/machine-learning-models
- https://www.mathworks.com/discovery/machine-learning-models.html
- https://blog.skillfactory.ru/glossary/keras/