As the climate changes, predicting the weather becomes ever more important for businesses. Since the weather depends on a lot of different factors, you will want to run a lot of experiments to determine what the best approach is to predict the weather. In this project, you will run experiments for different regression models predicting the mean temperature, using a combination of `sklearn` and `MLflow`.

You will be working with data stored in `london_weather.csv`, which contains the following columns:

- **date** - recorded date of measurement - (**int**)
- **cloud_cover** - cloud cover measurement in oktas - (**float**)
- **sunshine** - sunshine measurement in hours (hrs) - (**float**)
- **global_radiation** - irradiance measurement in Watt per square meter (W/m2) - (**float**)
- **max_temp** - maximum temperature recorded in degrees Celsius (°C) - (**float**)
- **mean_temp** - mean temperature in degrees Celsius (°C) - (**float**)
- **min_temp** - minimum temperature recorded in degrees Celsius (°C) - (**float**)
- **precipitation** - precipitation measurement in millimeters (mm) - (**float**)
- **pressure** - pressure measurement in Pascals (Pa) - (**float**)
- **snow_depth** - snow depth measurement in centimeters (cm) - (**float**)

In [6]:
```
import pandas as pd
import numpy as np
import mlflow
import mlflow.sklearn
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

Exploratory Data Analysis

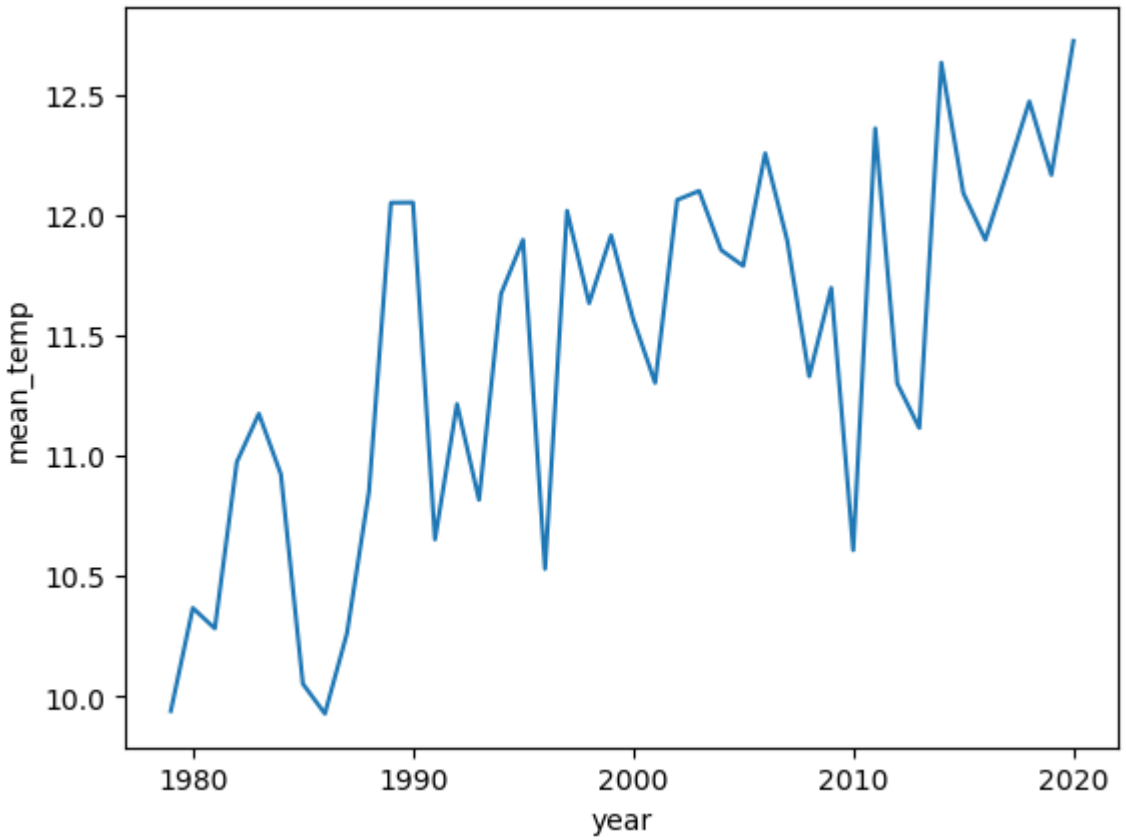
In [7]:
```
weather = pd.read_csv('london_weather.csv')
weather.info()

weather['date'] = pd.to_datetime(weather['date'], format='%Y%m%d')
weather['year'] = weather['date'].dt.year
weather['month'] = weather['date'].dt.month
weather_metrics = ['cloud_cover', 'sunshine', 'global_radiation', 'max_temp', 'mean_temp', 'min_temp', 'pre
weather_per_month = weather.groupby(['year', 'month'], as_index = False)[weather_metrics].mean()
```

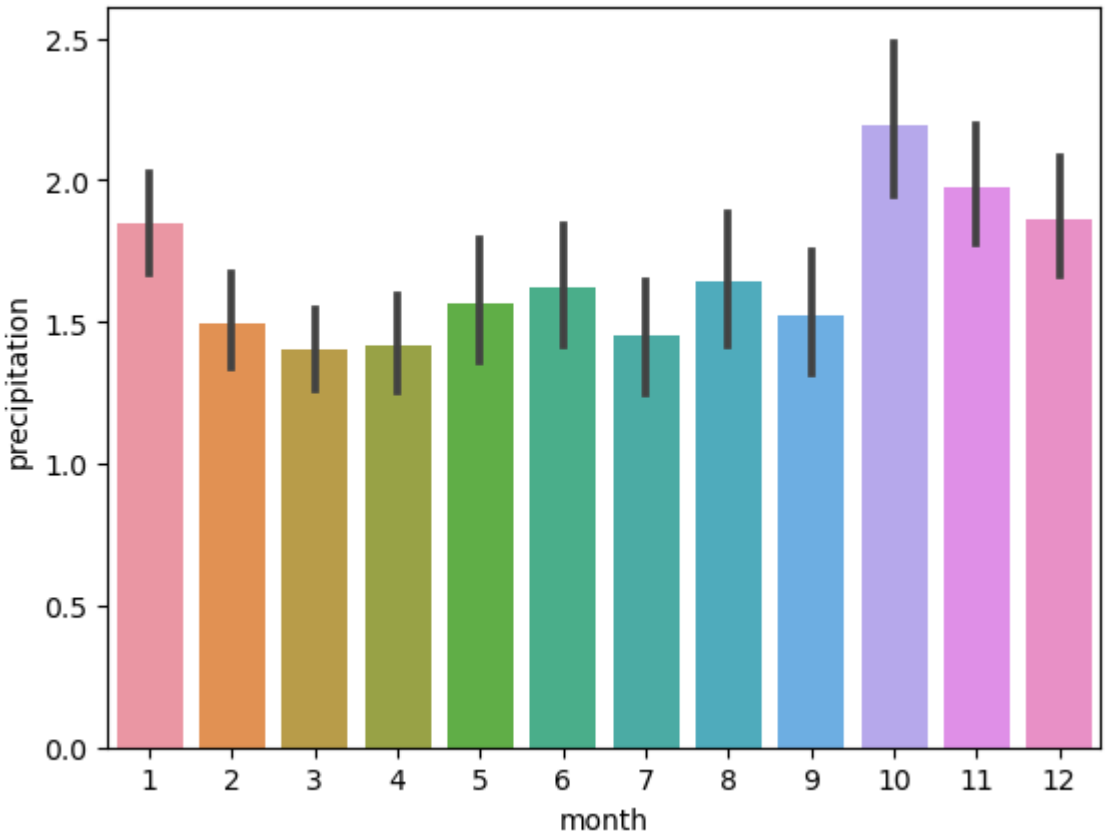
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15341 entries, 0 to 15340
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   date              15341 non-null  int64  
 1   cloud_cover       15322 non-null  float64
 2   sunshine          15341 non-null  float64
 3   global_radiation  15322 non-null  float64
 4   max_temp          15335 non-null  float64
 5   mean_temp         15305 non-null  float64
 6   min_temp          15339 non-null  float64
 7   precipitation     15335 non-null  float64
 8   pressure          15337 non-null  float64
 9   snow_depth        13900 non-null  float64
dtypes: float64(9), int64(1)
memory usage: 1.2 MB
```

Data Visualization

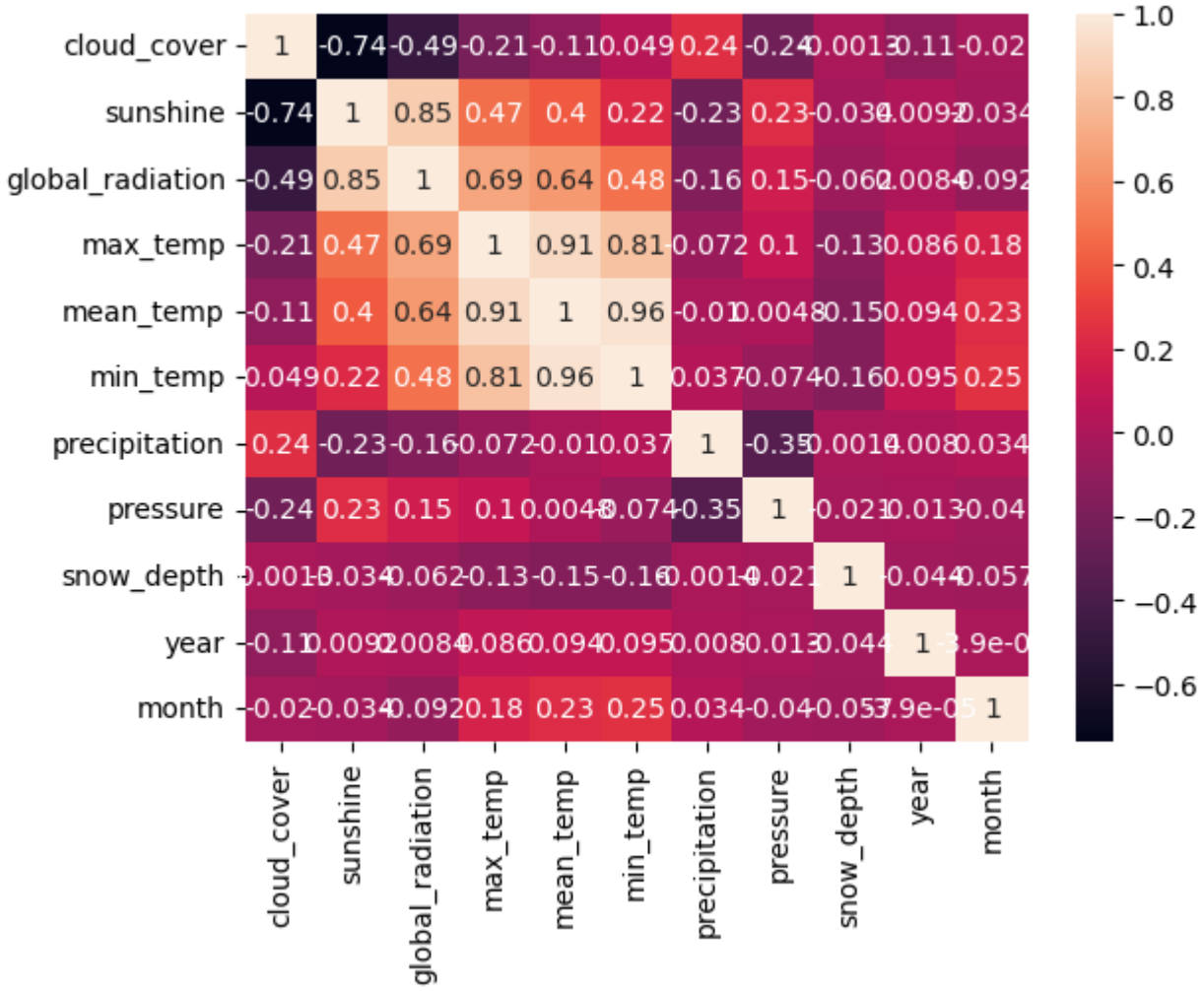
In [8]:
```
sns.lineplot(x="year", y="mean_temp", data=weather_per_month, ci=None)
plt.show()
sns.barplot(x='month', y='precipitation', data=weather)
plt.show()
sns.heatmap(weather.corr(), annot=True)
plt.show()
```

```
C:\Users\pc\AppData\Local\Temp\ipykernel_13248\3721478828.py:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.lineplot(x="year", y="mean_temp", data=weather_per_month, ci=None)
```


12.5
12.0
11.5
11.0
10.5
10.0
mean_temp
1980
1990
2000
2010
2020
year




```
C:\Users\pc\AppData\Local\Temp\ipykernel_13248\3721478828.py:5: FutureWarning: The default value of numeric_only in DataFrame.corr is depr
ecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warn
ing.
  sns.heatmap(weather.corr(), annot=True)
```



Feature Selection

In [9]:
```
feature_selection = ['month', 'cloud_cover', 'sunshine', 'precipitation', 'pressure', 'global_radiation']
target_var = 'mean_temp'
weather = weather.dropna(subset=['mean_temp'])
```

Preprocessing

In [10]:

```
def preprocess_df(df, feature_selection, target_var):
    """
    Split dataframe into X and y, and train and test consecutively. Then impute and scale both train and te
    """
    # Subset the data
    X = df[feature_selection]
    y = df[target_var]

    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)

    # Impute missing values
    imputer = SimpleImputer(strategy="mean")
    # Fit on the training data
    X_train = imputer.fit_transform(X_train)
    # Transform on the test data
    X_test  = imputer.transform(X_test)

    # Scale the data
    scaler = StandardScaler()
    # Fit on the training data
    X_train = scaler.fit_transform(X_train)
    # Transform on the test data
    X_test = scaler.transform(X_test)

    return X_train, X_test, y_train, y_test
```

```
X_train, X_test, y_train, y_test = preprocess_df(weather, feature_selection, target_var)
```

Machine Learning Training and Evaluation

In [11]:

```
# Import necessary libraries
import mlflow
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Predict on the test set and evaluate performance
def predict_and_evaluate(model, x_test, y_test):
    """
    Predict values from test set, calculate and return the root mean squared error.
    """
    y_pred = model.predict(x_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    return rmse

# Create an experiment if it does not exist
EXPERIMENT_NAME = "weather_prediction"
try:
    EXPERIMENT_ID = mlflow.create_experiment(EXPERIMENT_NAME)
except:
    EXPERIMENT_ID = mlflow.get_experiment_by_name(EXPERIMENT_NAME).experiment_id

# Predict, evaluate, and log the parameters and metrics of the models
for idx, depth in enumerate([1, 2, 5, 10, 20]):
    parameters = {
        'max_depth': depth
    }
```

```
        run_name = f"run_{idx}"
        with mlflow.start_run(experiment_id=EXPERIMENT_ID, run_name=run_name):
            # Create models
            lin_reg = LinearRegression().fit(X_train, y_train)
            tree_reg = DecisionTreeRegressor(random_state=42, max_depth=depth).fit(X_train, y_train)
            forest_reg = RandomForestRegressor(random_state=42, max_depth=depth).fit(X_train, y_train)
            # Log models
            mlflow.sklearn.log_model(lin_reg, "lin_reg")
            mlflow.sklearn.log_model(tree_reg, "tree_reg")
            mlflow.sklearn.log_model(forest_reg, "forest_reg")
            # Evaluate performance
            lin_reg_rmse = predict_and_evaluate(lin_reg, X_test, y_test)
            tree_reg_rmse = predict_and_evaluate(tree_reg, X_test, y_test)
            forest_reg_rmse = predict_and_evaluate(forest_reg, X_test, y_test)
            # Log performance
            mlflow.log_param("max_depth", depth)
            mlflow.log_metric("rmse_lr", lin_reg_rmse)
            mlflow.log_metric("rmse_tr", tree_reg_rmse)
            mlflow.log_metric("rmse_fr", forest_reg_rmse)

    # Search the runs for the experiment's results
    experiment_results = mlflow.search_runs(experiment_ids=[EXPERIMENT_ID])
    experiment_results
```

```
E:\Anaconda Nav\lib\site-packages\_distutils_hack\__init__.py:33: UserWarning: Setuptools is replacing distutils.
  warnings.warn("Setuptools is replacing distutils.")
```

	run_id	experiment_id	status	artifact_uri	start_time
0	f4df7be94fac40bbac17111729751ddb	684370974106002345	FINISHED	file:///C:/Users/pc/Downloads/mlruns/684370974...	2023-06-13 11:43:09.572000+00:00
1	ab131d04651142ba8ddee5ea8766f135	684370974106002345	FINISHED	file:///C:/Users/pc/Downloads/mlruns/684370974...	2023-06-13 11:42:08.841000+00:00
2	6408f721c9d646abb1f37437bc75d11e	684370974106002345	FINISHED	file:///C:/Users/pc/Downloads/mlruns/684370974...	2023-06-13 11:41:21.405000+00:00
3	25e585b12ddc472aba5ed936e3b0cbb4	684370974106002345	FINISHED	file:///C:/Users/pc/Downloads/mlruns/684370974...	2023-06-13 11:40:42.473000+00:00
4	8dd1e7ca110d4523aa86903188d4c17c	684370974106002345	FINISHED	file:///C:/Users/pc/Downloads/mlruns/684370974...	2023-06-13 11:39:43.710000+00:00