pjoject

October 10, 2023

[19]:
```
import pandas as pd
import json
df = pd.read_excel('data for ml.xlsx')
df.head()
```

[19]:

	Year	m	d	DATE	WEEK	Precipitation amount (mm)	Snow depth (cm)	Air temperature (degC)	Maximum temperature (degC)	Minimum temperature (degC)
0	1960	1	1	1960-01-01	1	-1.0	29	-1.5	0.1	-7.6
1	1960	1	2	1960-01-02	1	-1.0	28	0.3	1.3	-0.5
2	1960	1	3	1960-01-03	1	0.1	29	0.0	1.1	-0.6
3	1960	1	4	1960-01-04	2	0.0	29	-0.2	0.1	-1
4	1960	1	5	1960-01-05	2	17.6	29	0.4	1.9	-0.5

[20]:
```
# Initialize variables to track the year and whether snow has been recorded
current_year = None
week_number = []
year_list = []
# Iterate through the DataFrame to check for snow
for index, row in df.iterrows():
    year = row['Year']
    week = row['WEEK']
    snow_depth_str = row['Snow depth (cm)']

    # Check if we are in a new year
```

```
    if year != current_year:
        current_year = year
        snow_recorded = False
    # Convert the snow depth to a numeric value (float)
    try:
        snow_depth = float(snow_depth_str)
    except ValueError:
        # Handle the case where the value is not a valid number
        snow_depth = -1  # You can change this default value as needed
    # Check if the week number is in the specified range (40-55) and snow has␣
 ↪not been recorded yet for the year
    if 30 <= week <= 55 and snow_depth > 0 and not snow_recorded:
        week_number.append(week)
        snow_recorded = True
        year_list.append(current_year)
df = df[df['Year'].isin(year_list)]
# Print the list of week numbers with the first snow in each year
print(week_number)
```

```
[47, 49, 47, 47, 48, 47, 48, 50, 44, 44, 45, 45, 45, 41, 48, 44, 45, 47, 49, 44,
43, 47, 47, 47, 48, 48, 49, 46, 45, 48, 47, 47, 43, 47, 47, 47, 48, 45, 45, 47,
53, 45, 44, 43, 47, 48, 45, 46, 48, 51, 47, 49, 47, 45, 44, 51, 45, 50, 47]
```

[21]:
```
print(len(week_number))
print(year_list)
```

```
59
[1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972,
1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985,
1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998,
1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2012,
2014, 2016, 2017, 2018, 2019, 2021, 2022]
```

[22]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# Read your DataFrame containing 'Precipitation amount (mm)', 'Maximum␣
 ↪temperature (degC)', 'Minimum temperature (degC)'
# Make sure you have loaded the data correctly into 'df'

# Read your 'week_number' list (the target)
```

```
# Assuming 'week_number' is a list containing the number of first snow week for each year
# You may need to adjust this code to read your actual data

# Define the features and target
features = ['Precipitation amount (mm)', 'Maximum temperature (degC)', 'Minimum temperature (degC)']

# Fill missing values with the mean for numeric columns
#df[features] = df[features].apply(pd.to_numeric, errors='coerce')
#df.fillna(df.mean(), inplace=True)
# Create a DataFrame for the features (35 days of data for each year)
X = []

# Assuming you have data for 63 years from 1960 to 2022
for year in range(1960, 2023):
    # Extract data for weeks 34 to 38 for each year
    # Replace this with your actual data filtering logic
    # This assumes df contains data for each day
    year_data = df[(df['Year'] == year) & (df['WEEK'] >= 34) & (df['WEEK'] <= 38)]
    # Check if there are 35 days of data for this year
    if len(year_data) == 35:
        #print(year)
        # Append the relevant features to X
        X.append(year_data[features].values)
```

[23]:
```
# Convert X to a NumPy array
X = np.array(X)

# Combine the scaled data back into a single array
X = X.reshape((59, 35 * len(features)))



# Read your 'week_number' list (the target)
# Assuming 'week_number' is a list containing the number of first snow week for each year
# You may need to adjust this code to read your actual data

# Create a DataFrame for the target variable 'week_number'

# Convert 'week_number' to a NumPy array
y = np.array(week_number)

# Feature scaling
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

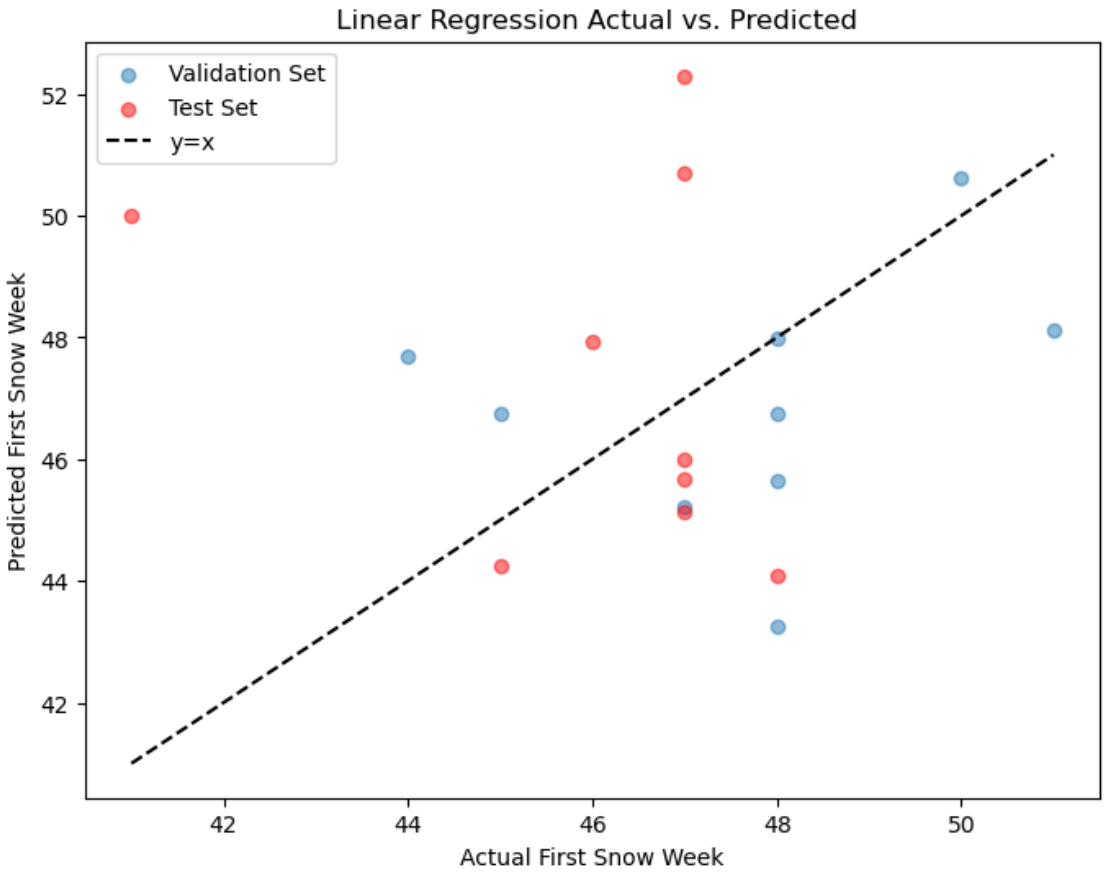
[24]:
```
# Split the data into training, validation, and testing sets
X_train, X_temp, y_train, y_temp = train_test_split(X_scaled, y, test_size=0.3,␣
↪random_state=42)
X_valid, X_test, y_valid, y_test = train_test_split(X_temp, y_temp, test_size=0.
↪5, random_state=42)
```

[25]:
```
# Initialize and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict on the validation set
y_valid_pred = model.predict(X_valid)

# Calculate Mean Squared Error (MSE) on the validation set
mse = mean_squared_error(y_valid, y_valid_pred)
print(f"Validation MSE: {mse}")

# # Visualize the predicted vs. actual values on the validation set
# plt.scatter(y_valid, y_valid_pred)
# plt.xlabel("Actual First Snow Week")
# plt.ylabel("Predicted First Snow Week")
# plt.title("Validation Set: Actual vs. Predicted")
# plt.show()

# Predict on the test set
y_test_pred = model.predict(X_test)



# Calculate MSE on the test set
test_mse = mean_squared_error(y_test, y_test_pred)
print(f"Test MSE: {test_mse}")
```

```
Validation MSE: 6.4566442916744045
Test MSE: 16.5245214617304
```

[26]:
```
import matplotlib.pyplot as plt
import numpy as np
# Create a scatter plot for the validation set
plt.figure(figsize=(8, 6))
plt.scatter(y_valid, y_valid_pred, label="Validation Set", alpha=0.5)
plt.scatter(y_test, y_test_pred, label="Test Set", alpha=0.5, c='r')
plt.xlabel("Actual First Snow Week")
plt.ylabel("Predicted First Snow Week")
```

```
plt.title(" Linear Regression Actual vs. Predicted")
plt.legend()

# Add a line y=x to the plot
x_line = np.linspace(min(y_valid.min(), y_test.min()), max(y_valid.max(),␣
 ↪y_test.max()), 100)
plt.plot(x_line, x_line, color='k', linestyle='--', label='y=x')

plt.legend()
plt.show()
```

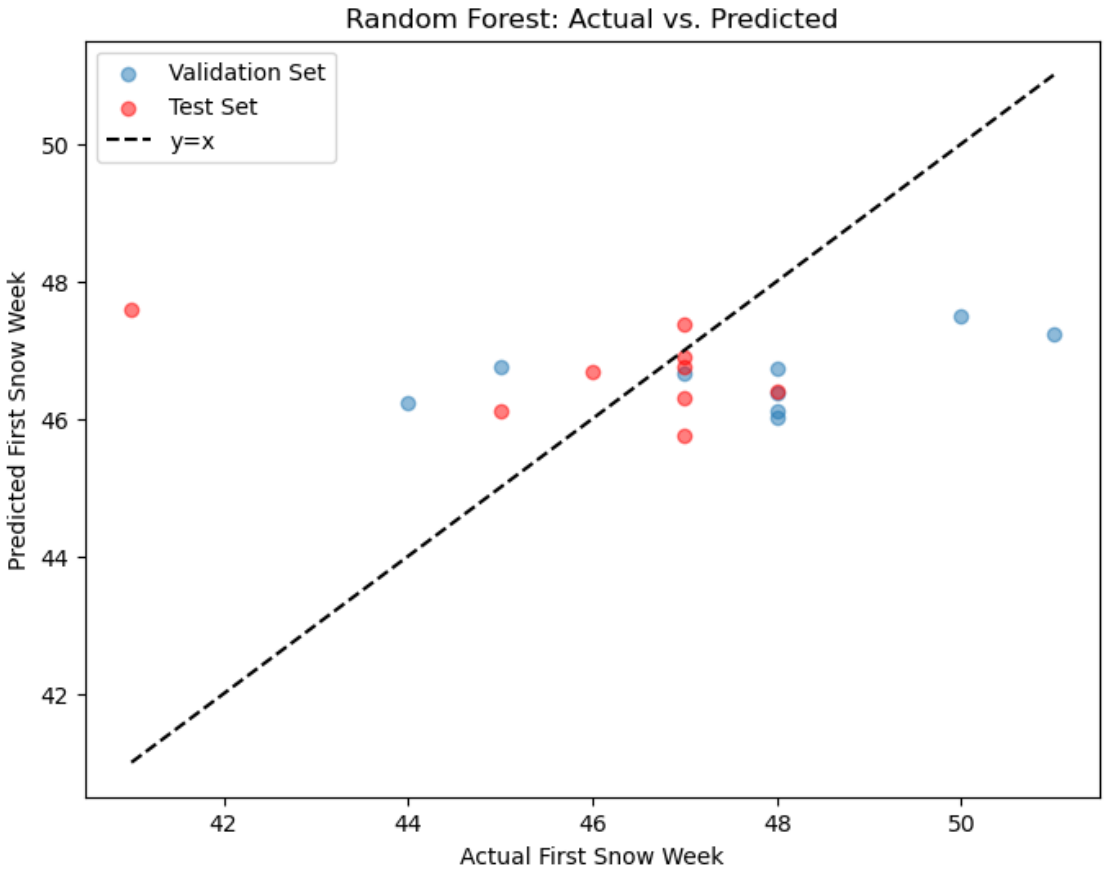


[27]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Create and configure the random forest regressor
rf_model = RandomForestRegressor(n_estimators=100, max_depth=10,␣
 ↪random_state=42)

# Train the model on the training data
```

```
rf_model.fit(X_train, y_train)

# Make predictions on the validation set
y_pred_valid = rf_model.predict(X_valid)

# Evaluate the model on the validation set
mse = mean_squared_error(y_valid, y_pred_valid, squared=False)
print(f"Validation MSE: {mse}")

# Make predictions on the test set
y_pred_test = rf_model.predict(X_test)

# Evaluate the model on the test set
test_mse = mean_squared_error(y_test, y_pred_test, squared=False)
print(f"Test MSE: {test_mse}")
```

```
Validation MSE: 2.1182041660068758
Test MSE: 2.3555537211732913
```

[28]:
```
# Create a scatter plot for the validation set
plt.figure(figsize=(8, 6))
plt.scatter(y_valid, y_pred_valid, label="Validation Set", alpha=0.5)
plt.scatter(y_test, y_pred_test, label="Test Set", alpha=0.5, c='r')
plt.xlabel("Actual First Snow Week")
plt.ylabel("Predicted First Snow Week")
plt.title("Random Forest: Actual vs. Predicted")
plt.legend()

# Add a line y=x to the plot
x_line = np.linspace(min(y_valid.min(), y_test.min()), max(y_valid.max(),␣
 ↪y_test.max()), 100)
plt.plot(x_line, x_line, color='k', linestyle='--', label='y=x')

plt.legend()
plt.show()
```


Random Forest: Actual vs. Predicted
Validation Set
Test Set
y=x
Predicted First Snow Week
Actual First Snow Week
42
44
46
48
50

[ ]:

[ ]: