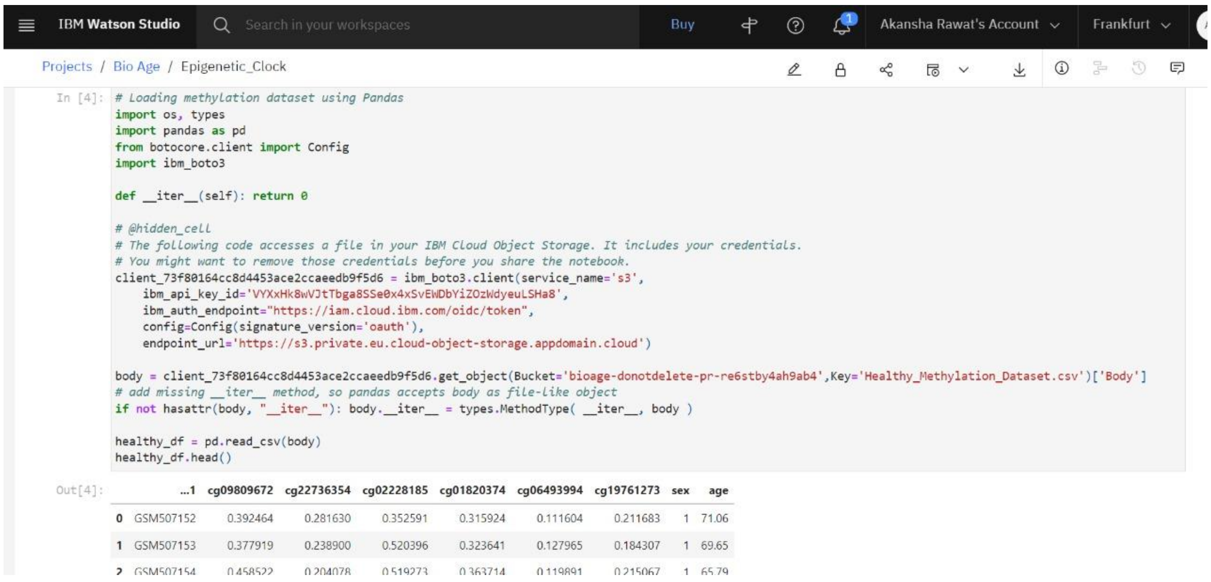
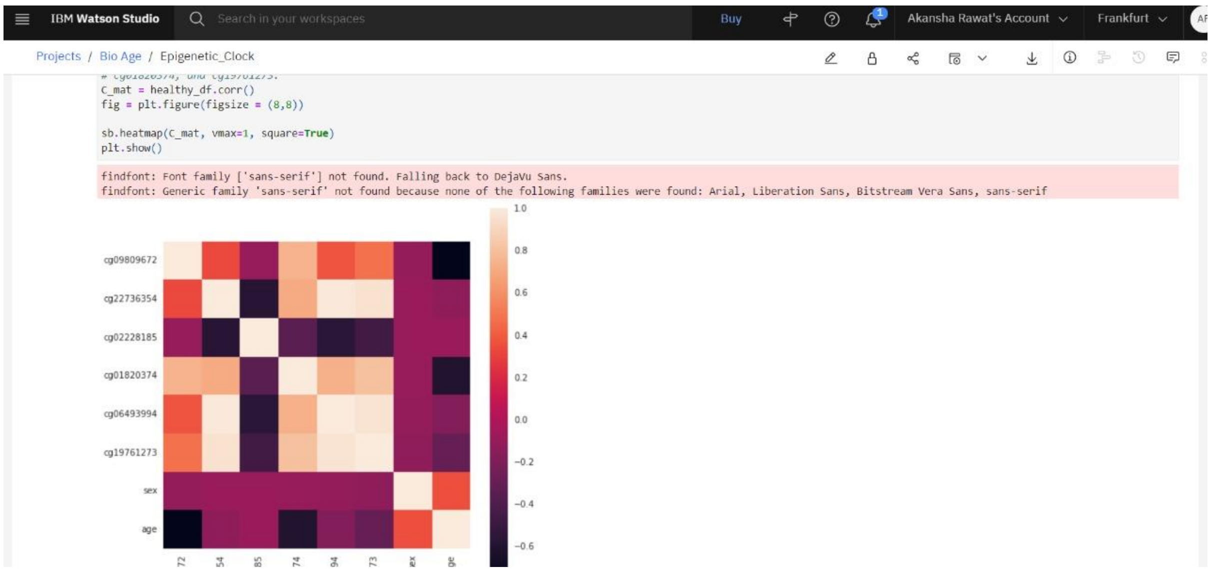
Results


IBM Watson Studio
Search in your workspaces
Buy
Akansha Rawat's Account
Frankfurt
Projects / Bio Age / Epigenetic_Clock

```
In [4]: # Loading methylation dataset using Pandas
        import os, types
        import pandas as pd
        from botocore.client import Config
        import ibm_boto3

        def __iter__(self): return 0

        # @hidden_cell
        # The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
        # You might want to remove those credentials before you share the notebook.
        client_73f80164cc8d4453ace2ccaeedb9f5d6 = ibm_boto3.client(service_name='s3',
            ibm_api_key_id='VYXxHk8wVJtTbga8SSe0x4xSvEWDbYiZOzWdyeuLSHa8',
            ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
            config=Config(signature_version='oauth'),
            endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')

        body = client_73f80164cc8d4453ace2ccaeedb9f5d6.get_object(Bucket='bioage-donotdelete-pr-re6stby4ah9ab4',Key='Healthy_Methylation_Dataset.csv')['Body']
        # add missing __iter__ method, so pandas accepts body as file-like object
        if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

        healthy_df = pd.read_csv(body)
        healthy_df.head()
```

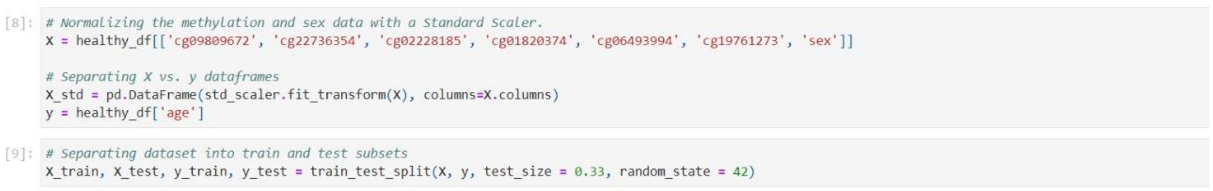
Out[4]:

	...1	cg09809672	cg22736354	cg02228185	cg01820374	cg06493994	cg19761273	sex	age
0	GSM507152	0.392464	0.281630	0.352591	0.315924	0.111604	0.211683	1	71.06
1	GSM507153	0.377919	0.238900	0.520396	0.323641	0.127965	0.184307	1	69.65
2	GSM507154	0.458522	0.204078	0.519273	0.363714	0.119891	0.215067	1	65.79

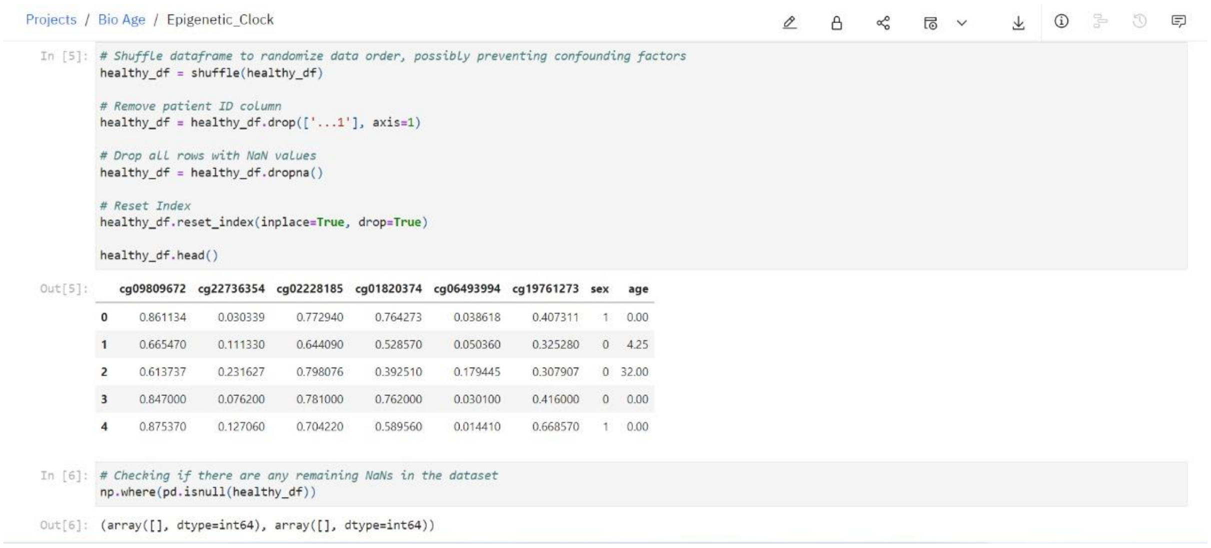


```
C_mat = healthy_df.corr()
fig = plt.figure(figsize = (8,8))

sb.heatmap(C_mat, vmax=1, square=True)
plt.show()
```

```
findfont: Font family ['sans-serif'] not found. Falling back to DejaVu Sans.
findfont: Generic family 'sans-serif' not found because none of the following families were found: Arial, Liberation Sans, Bitstream Vera Sans, sans-serif
```



```
[8]: # Normalizing the methylation and sex data with a Standard Scaler.
     X = healthy_df[['cg09809672', 'cg22736354', 'cg02228185', 'cg01820374', 'cg06493994', 'cg19761273', 'sex']]

     # Separating X vs. y dataframes
     X_std = pd.DataFrame(std_scaler.fit_transform(X), columns=X.columns)
     y = healthy_df['age']

[9]: # Separating dataset into train and test subsets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 42)
```

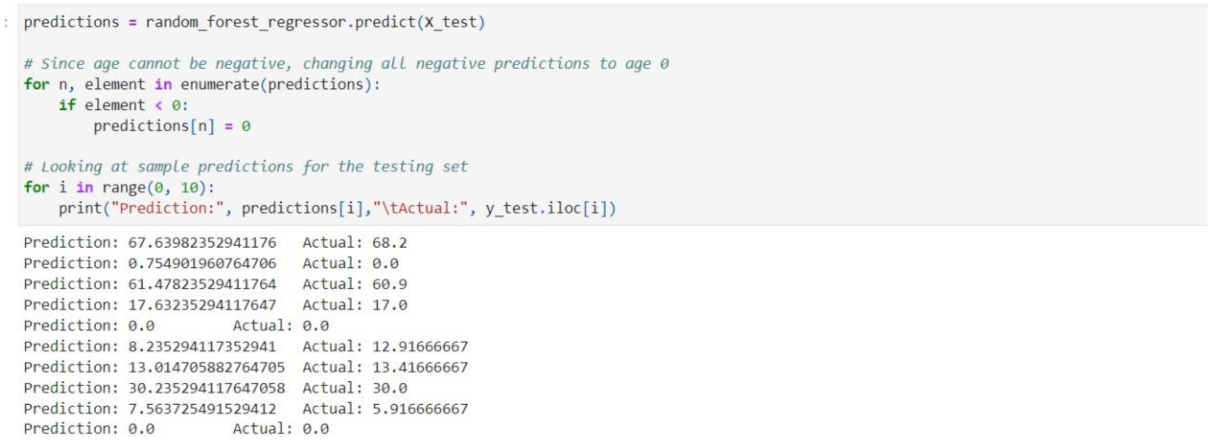


```
In [5]: # Shuffle dataframe to randomize data order, possibly preventing confounding factors
        healthy_df = shuffle(healthy_df)

        # Remove patient ID column
        healthy_df = healthy_df.drop(['...1'], axis=1)

        # Drop all rows with NaN values
        healthy_df = healthy_df.dropna()

        # Reset Index
        healthy_df.reset_index(inplace=True, drop=True)

        healthy_df.head()
```

Out[5]:

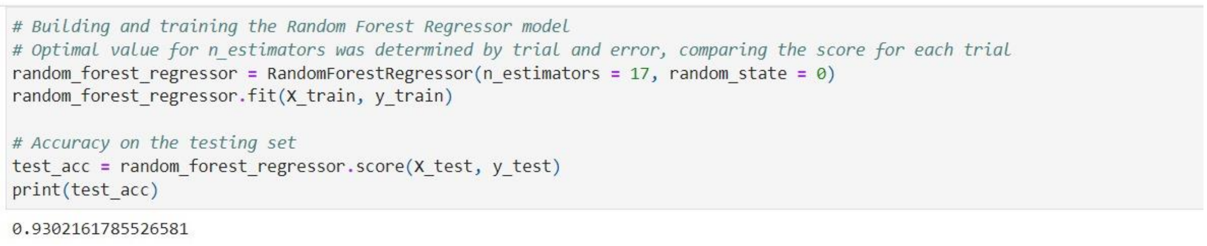
	cg09809672	cg22736354	cg02228185	cg01820374	cg06493994	cg19761273	sex	age
0	0.861134	0.030339	0.772940	0.764273	0.038618	0.407311	1	0.00
1	0.665470	0.111330	0.644090	0.528570	0.050360	0.325280	0	4.25
2	0.613737	0.231627	0.798076	0.392510	0.179445	0.307907	0	32.00
3	0.847000	0.076200	0.781000	0.762000	0.030100	0.416000	0	0.00
4	0.875370	0.127060	0.704220	0.589560	0.014410	0.668570	1	0.00

```
In [6]: # Checking if there are any remaining NaNs in the dataset
        np.where(pd.isnull(healthy_df))

Out[6]: (array([], dtype=int64), array([], dtype=int64))
```



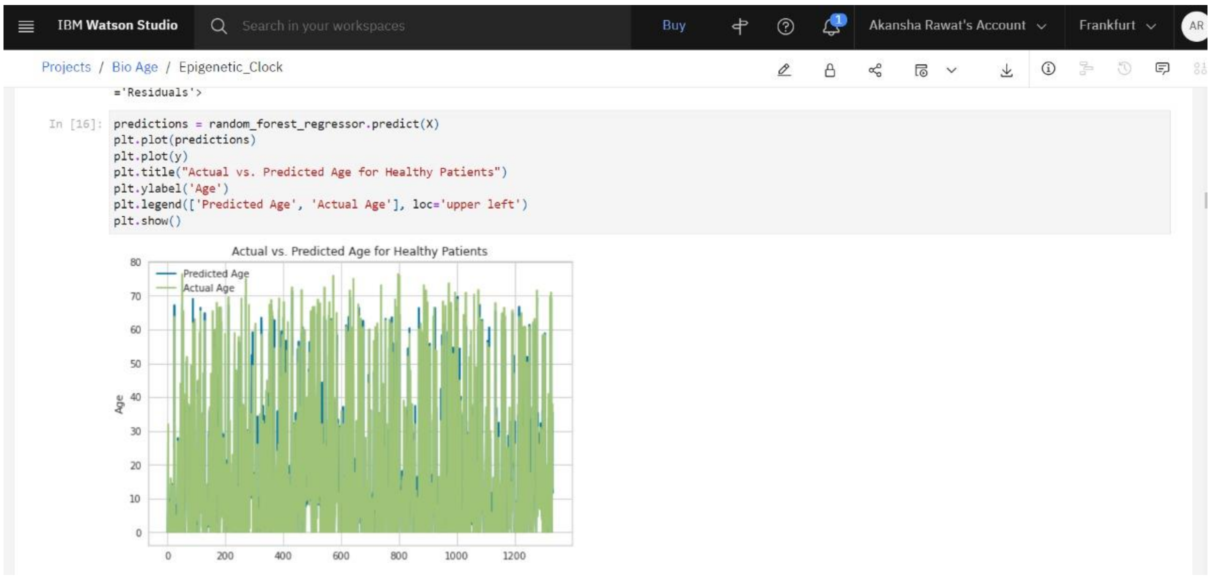
```
predictions = random_forest_regressor.predict(X_test)

# Since age cannot be negative, changing all negative predictions to age 0
for n, element in enumerate(predictions):
    if element < 0:
        predictions[n] = 0

# Looking at sample predictions for the testing set
for i in range(0, 10):
    print("Prediction:", predictions[i],"\tActual:", y_test.iloc[i])
```

```
Prediction: 67.63982352941176   Actual: 68.2
Prediction: 0.754901960764706   Actual: 0.0
Prediction: 61.47823529411764   Actual: 60.9
Prediction: 17.63235294117647   Actual: 17.0
Prediction: 0.0         Actual: 0.0
Prediction: 8.235294117352941   Actual: 12.91666667
Prediction: 13.014705882764705  Actual: 13.41666667
Prediction: 30.235294117647058  Actual: 30.0
Prediction: 7.563725491529412   Actual: 5.916666667
Prediction: 0.0         Actual: 0.0
```

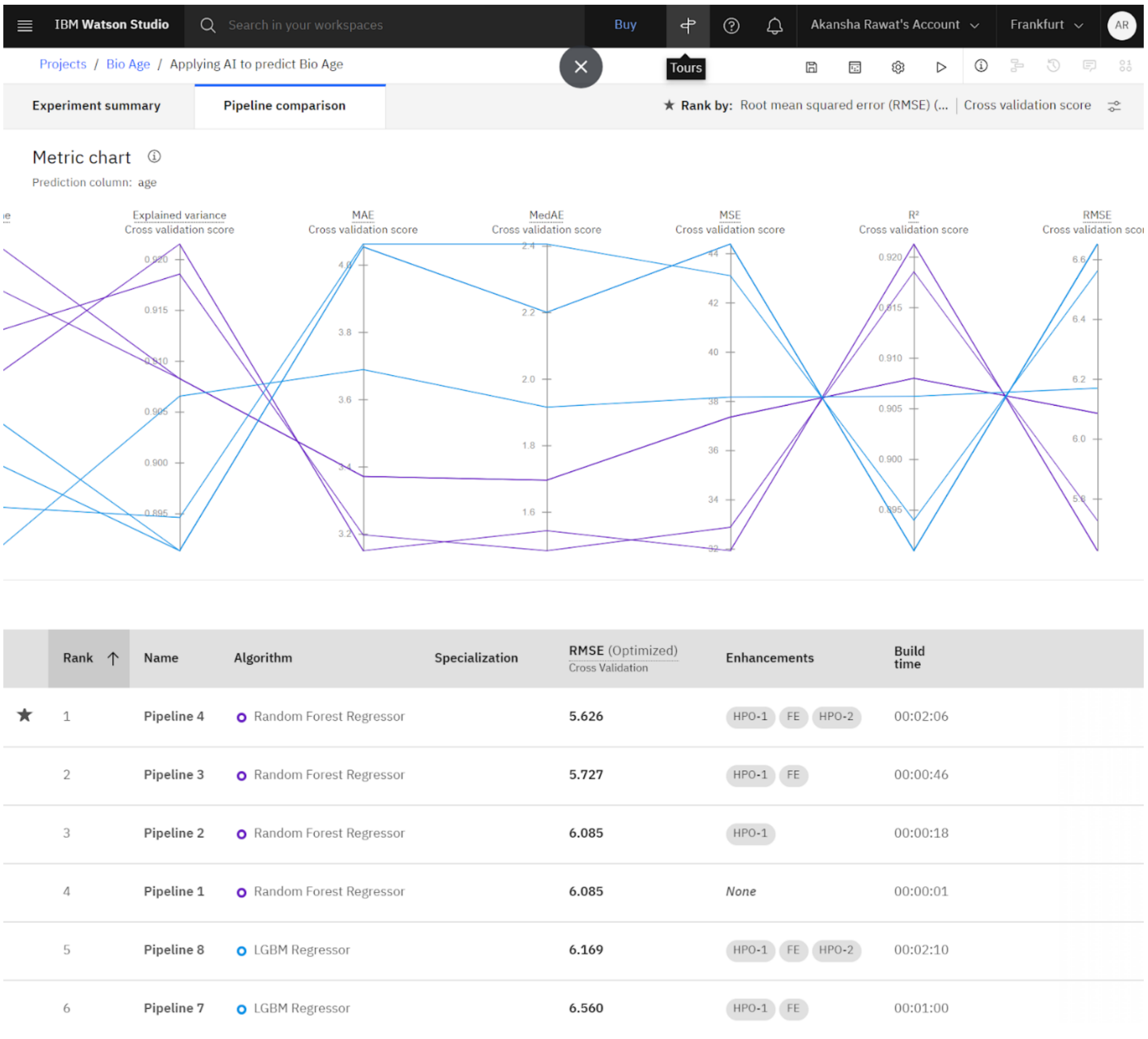


```
# Building and training the Random Forest Regressor model
# Optimal value for n_estimators was determined by trial and error, comparing the score for each trial
random_forest_regressor = RandomForestRegressor(n_estimators = 17, random_state = 0)
random_forest_regressor.fit(X_train, y_train)

# Accuracy on the testing set
test_acc = random_forest_regressor.score(X_test, y_test)
print(test_acc)
```

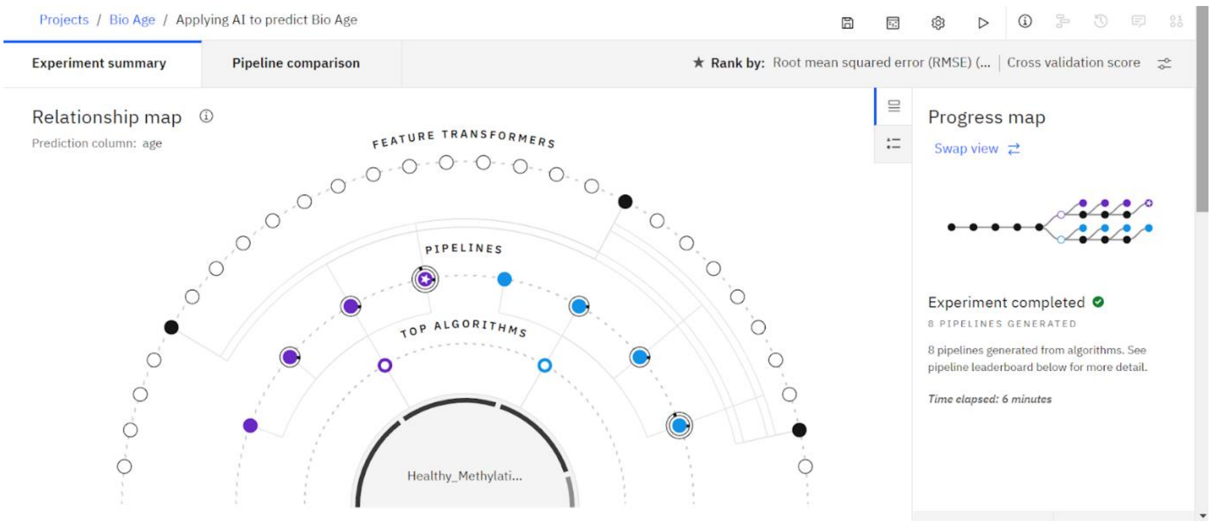
```
0.9302161785526581
```



```
='Residuals'>

In [16]: predictions = random_forest_regressor.predict(X)
         plt.plot(predictions)
         plt.plot(y)
         plt.title("Actual vs. Predicted Age for Healthy Patients")
         plt.ylabel('Age')
         plt.legend(['Predicted Age', 'Actual Age'], loc='upper left')
         plt.show()
```



	Rank	Name	Algorithm	Specialization	RMSE (Optimized) Cross Validation	Enhancements	Build time
★	1	Pipeline 4	Random Forest Regressor		5.626	HPO-1 FE HPO-2	00:02:06
	2	Pipeline 3	Random Forest Regressor		5.727	HPO-1 FE	00:00:46
	3	Pipeline 2	Random Forest Regressor		6.085	HPO-1	00:00:18
	4	Pipeline 1	Random Forest Regressor		6.085	*None*	00:00:01
	5	Pipeline 8	LGBM Regressor		6.169	HPO-1 FE HPO-2	00:02:10
	6	Pipeline 7	LGBM Regressor		6.560	HPO-1 FE	00:01:00

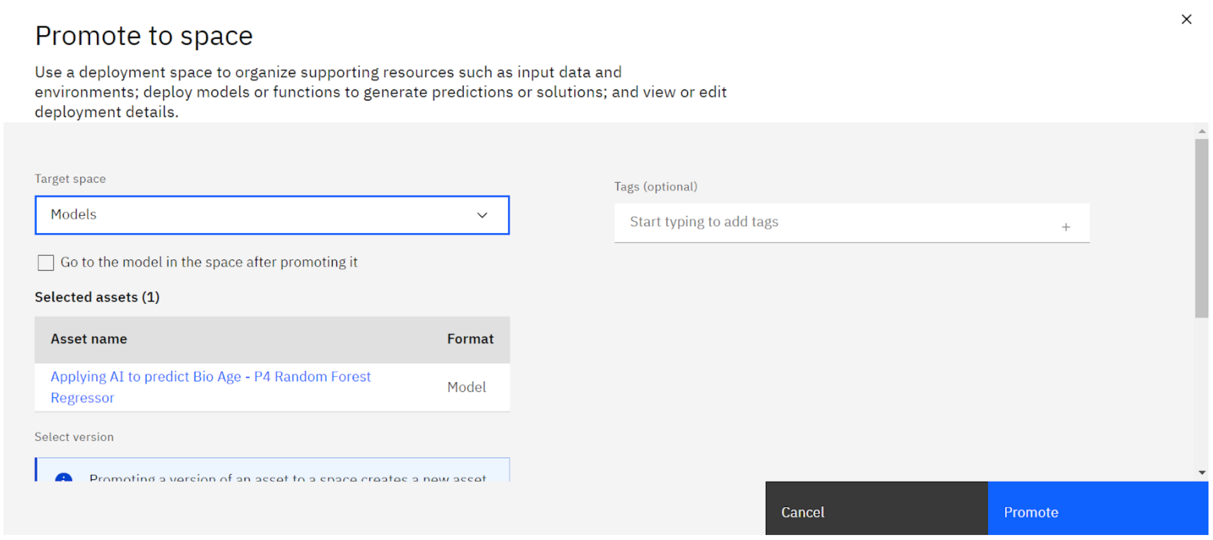

Projects / Bio Age / Applying AI to predict Bio Age

Experiment summary | Pipeline comparison | ★ Rank by: Root mean squared error (RMSE) (... | Cross validation score

Relationship map
Prediction column: age

Progress map
Swap view

Experiment completed
8 PIPELINES GENERATED
8 pipelines generated from algorithms. See pipeline leaderboard below for more detail.
Time elapsed: 6 minutes


Promote to space

Use a deployment space to organize supporting resources such as input data and environments; deploy models or functions to generate predictions or solutions; and view or edit deployment details.

Target space: Models

Tags (optional): Start typing to add tags

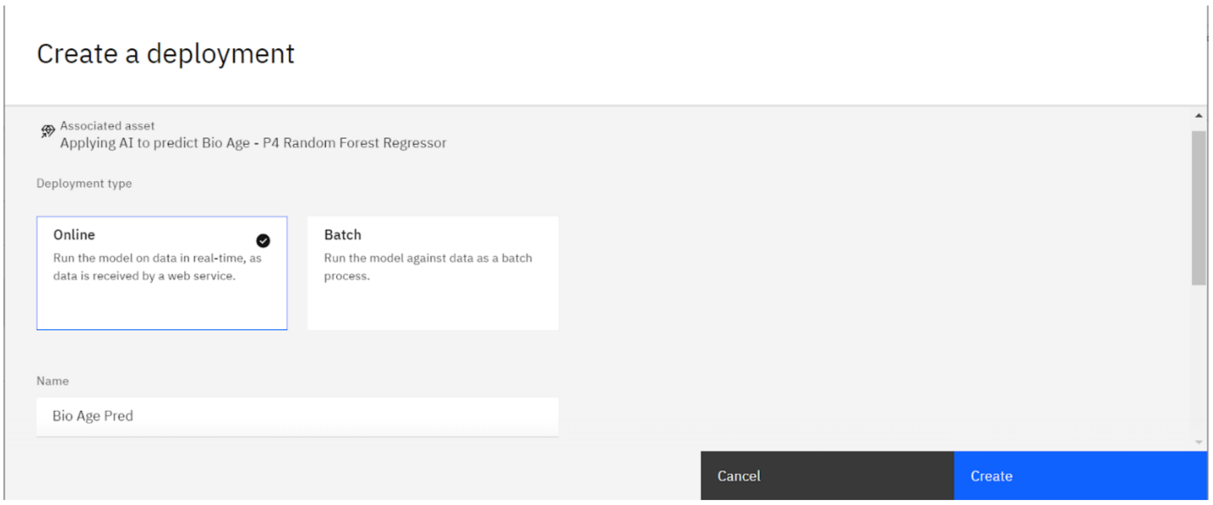
☐ Go to the model in the space after promoting it

Selected assets (1)

Asset name	Format
Applying AI to predict Bio Age - P4 Random Forest Regressor	Model

Select version

Promoting a version of an asset to a space creates a new asset


Create a deployment

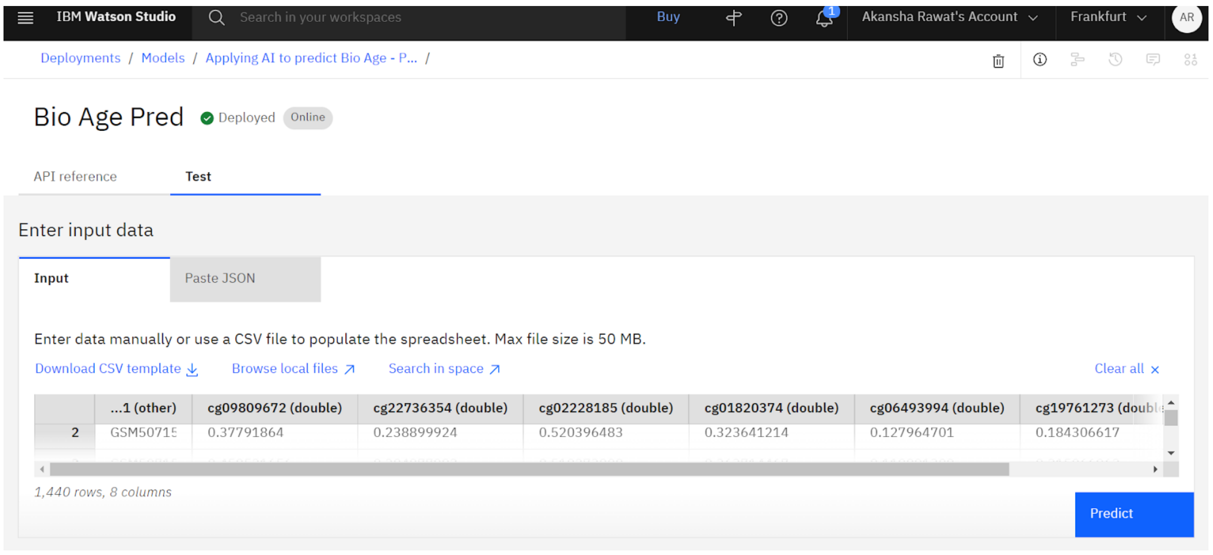
Associated asset
Applying AI to predict Bio Age - P4 Random Forest Regressor

Deployment type

[✓ Online] Run the model on data in real-time, as data is received by a web service.

Batch: Run the model against data as a batch process.

Name: Bio Age Pred



Deployments / Models / Applying AI to predict Bio Age - P... /

Bio Age Pred Deployed Online

API reference Test

Enter input data

Input Paste JSON

Enter data manually or use a CSV file to populate the spreadsheet. Max file size is 50 MB.

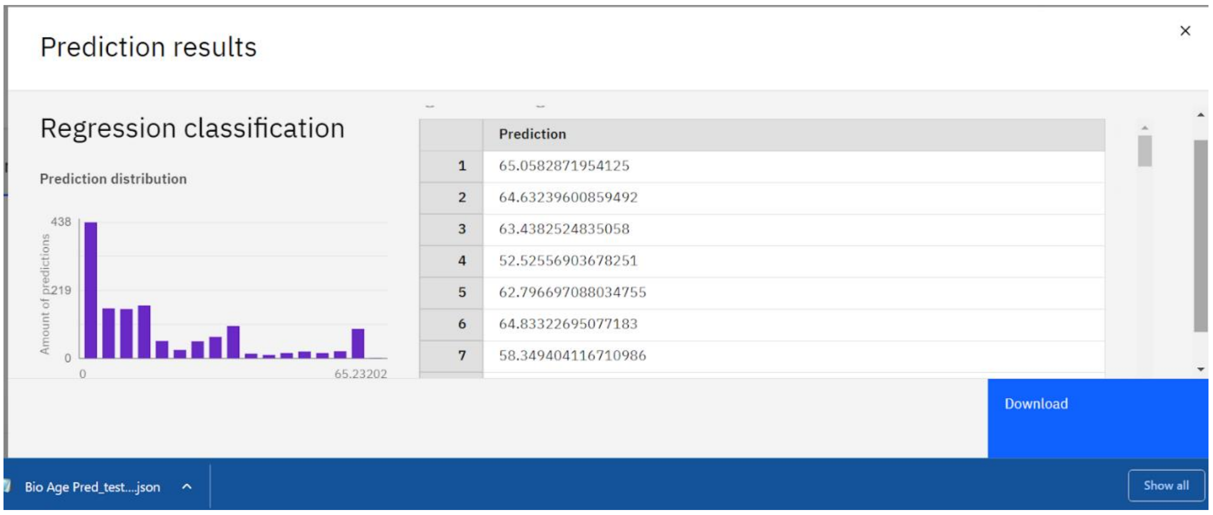
Download CSV template Browse local files Search in space Clear all

	...1 (other)	cg09809672 (double)	cg22736354 (double)	cg02228185 (double)	cg01820374 (double)	cg06493994 (double)	cg19761273 (doubl
2	GSM50715	0.37791864	0.238899924	0.520396483	0.323641214	0.127964701	0.184306617

1,440 rows, 8 columns

Predict

Browse local files to add your dataset file and click predict.



Prediction results

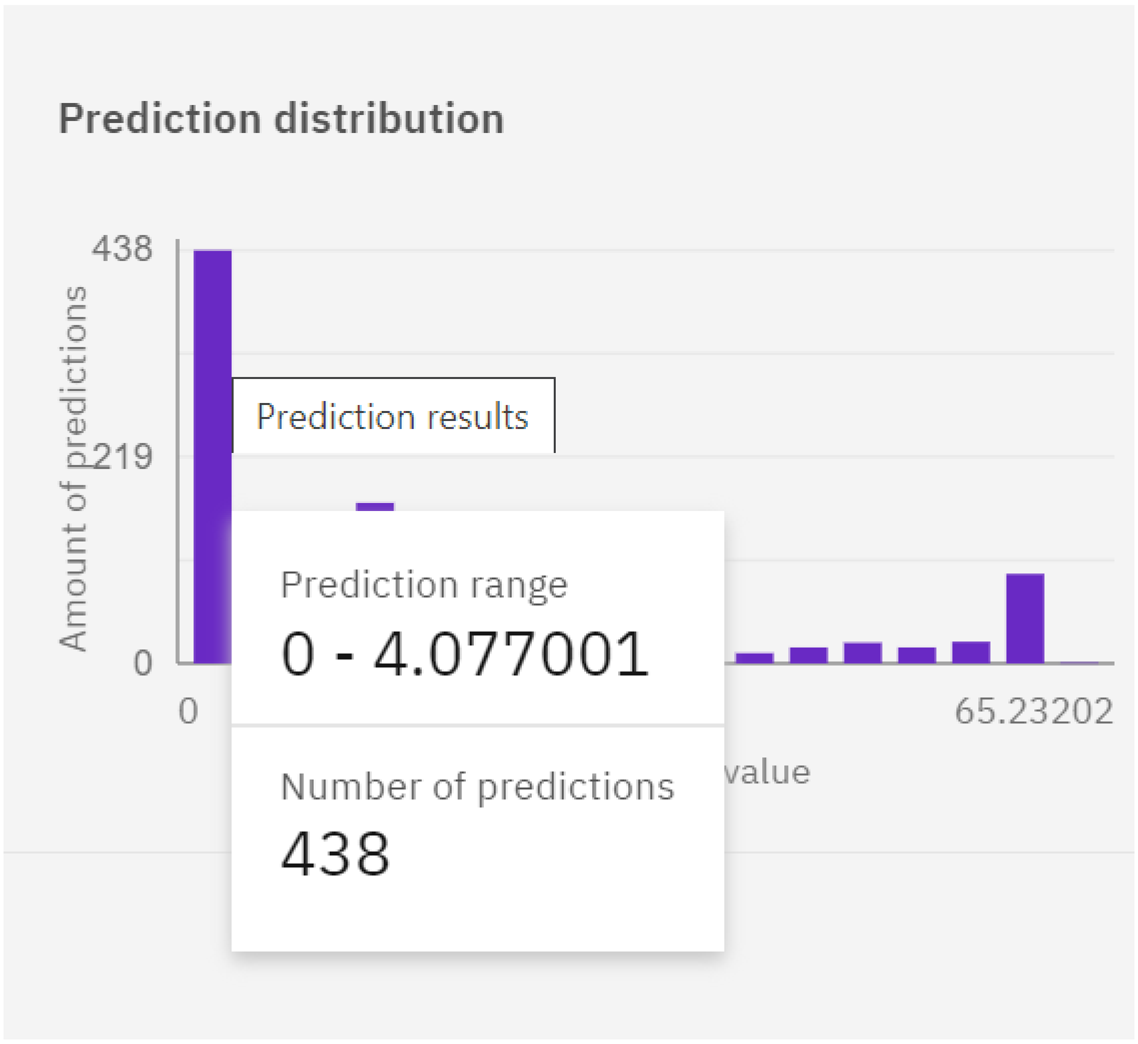
Regression classification

Prediction distribution

	Prediction
1	65.0582871954125
2	64.63239600859492
3	63.4382524835058
4	52.52556903678251
5	62.796697088034755
6	64.83322695077183
7	58.349404116710986

Download

After predicting, download the results as a .json file.



Each bar shows the range and estimated predictions.