MECE 6397: Project 2

Syam Evani
07/31/2025

Introduction

Table of Contents

1. Part 1: Timer Illustration
2. Part 2: Constructing DOE
3. Part 3: Full Factorial Output Modification
4. Part 4: Updating Random Forest Model
5. Part 5: Analysis & Recommendations

My code for this project is included in three ways:

- Embedded in portions with relevant discussion
- Uploaded as part of this deliverable on Canvas
- Available on a public github here: https://github.com/Sam-v6/mece-6397-doe/tree/main/project2

Part 1: Timer Illustration

I have packaged the provided code and ran it below to illustrate the use of the timer. Note it takes only 0.27 seconds to perform the sample code routine. Timers are excellent for gauging the computational effcicienis (or inefficiencies) as new features are added during a model's implementation. Note, I also have simply packaged this into a `timer.py` file as requested.

In [ ]:
```
# Standard imports
import os
import time

# Additional imports
import pandas as pd
import numpy as np
import itertools
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from itertools import combinations
```

```python
# Local imports
# None

#---------------------------------------------------------------------
# Timing examples
#---------------------------------------------------------------------
class TimerError(Exception):
    """A custom exception used to report errors in use of Timer class"""

## Illustration of a Class Function Measure Time Performance
class Timer:
    def __init__(self):
        self._start_time = None

    def start(self):
        """Start a new timer"""
        if self._start_time is not None:
            raise TimerError(f"Timer is running. Use .stop() to stop it")

        self._start_time = time.perf_counter()

    def stop(self):
        """Stop the timer, and report the elapsed time"""
        if self._start_time is None:
            raise TimerError(f"Timer is not running. Use .start() to start it")

        elapsed_time = time.perf_counter() - self._start_time
        elapsed_time_min = (time.perf_counter() - self._start_time)/60
        self._start_time = None
        print(f"Elapsed time: {elapsed_time:0.4f} seconds")
        print(f"Elapsed time: {elapsed_time_min:0.4f} minutes")

#---------------------------------------------------------------------
# Create design
#---------------------------------------------------------------------
# Define the factors and levels
factors = {
    'n_estimators': ['-', '+'],
    'max_depth': ['-', '+']
}

# Create a full factorial design
def full_factorial_design(factors):
    import itertools
    levels = list(factors.values())
    design = list(itertools.product(*levels))
    return pd.DataFrame(design, columns=factors.keys())

# Generate the design matrix
design_matrix = full_factorial_design(factors)
print(design_matrix)

#---------------------------------------------------------------------
# Run experiment
#---------------------------------------------------------------------
```

```python
# Import the time library
t = Timer()
t.start()

# Load the Iris dataset
X, y = load_iris(return_X_y=True)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Define the experiment configurations (2^2 design)
configurations = [
    {'n_estimators': 100, 'max_depth': 2},
    {'n_estimators': 100, 'max_depth': 10},
    {'n_estimators': 200, 'max_depth': 2},
    {'n_estimators': 200, 'max_depth': 10},
]

results = []

# Run the experiments
for config in configurations:
    clf = RandomForestClassifier(n_estimators=config['n_estimators'], max_depth=con
    clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)

    # Store the results
    results.append({
        'n_estimators': config['n_estimators'],
        'max_depth': config['max_depth'],
        'accuracy': accuracy
    })

t.stop()  # A few seconds later

# Convert results to a DataFrame for analysis
results_df = pd.DataFrame(results)

# Display the results
print(results_df)
```

```
  n_estimators max_depth
0            -         -
1            -         +
2            +         -
3            +         +
Elapsed time: 0.2649 seconds
Elapsed time: 0.0044 minutes
   n_estimators  max_depth  accuracy
0           100          2       1.0
1           100         10       1.0
2           200          2       1.0
3           200         10       1.0
```

Part 2: Creating five factor and two level DOE implementation

I've adjusted the provided code to include the 5 factors identified as well as tie them to the provided lower and higher levels. Note, some additional data was also provided but for now I'll use the original factor data to generate the combinations. In total we see 32 combinations between our 5 factors, each with 2 levels.

In [ ]:

```
# Standard imports
import os
import time

# Additional imports
import pandas as pd
import numpy as np
import itertools
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from itertools import combinations

# Local imports
# None

#---------------------------------------------------------------------
# Design matrix creation
#---------------------------------------------------------------------
# Define the factors and levels

# Provided values (yields accuracy 1.0 for all combos)
factors = {
    'n_estimators': [50, 100],
    'max_depth': [1, 20],
    'min_samples_split': [2, 6],
    'min_samples_leaf': [1, 5],
    'max_samples': [0.5, 0.8]
}

# New factors (yields accuracy 0.3 to 1.0)
# factors = {
#     'n_estimators': [5, 50],
#     'max_depth': [1, 10],
#     'min_samples_split': [2, 10],
#     'min_samples_leaf': [1, 10],
#     'max_samples': [0.1, 0.9]
# }

# Create a full factorial design
def full_factorial_design(factors):
    levels = list(factors.values())
```

```
    design = list(itertools.product(*levels))
    return pd.DataFrame(design, columns=factors.keys())

# Generate the design matrix
design_matrix = full_factorial_design(factors)
print(design_matrix)
```

```
    n_estimators  max_depth  min_samples_split  min_samples_leaf  max_samples
0             50          1                  2                 1          0.5
1             50          1                  2                 1          0.8
2             50          1                  2                 5          0.5
3             50          1                  2                 5          0.8
4             50          1                  6                 1          0.5
5             50          1                  6                 1          0.8
6             50          1                  6                 5          0.5
7             50          1                  6                 5          0.8
8             50         20                  2                 1          0.5
9             50         20                  2                 1          0.8
10            50         20                  2                 5          0.5
11            50         20                  2                 5          0.8
12            50         20                  6                 1          0.5
13            50         20                  6                 1          0.8
14            50         20                  6                 5          0.5
15            50         20                  6                 5          0.8
16           100          1                  2                 1          0.5
17           100          1                  2                 1          0.8
18           100          1                  2                 5          0.5
19           100          1                  2                 5          0.8
20           100          1                  6                 1          0.5
21           100          1                  6                 1          0.8
22           100          1                  6                 5          0.5
23           100          1                  6                 5          0.8
24           100         20                  2                 1          0.5
25           100         20                  2                 1          0.8
26           100         20                  2                 5          0.5
27           100         20                  2                 5          0.8
28           100         20                  6                 1          0.5
29           100         20                  6                 1          0.8
30           100         20                  6                 5          0.5
31           100         20                  6                 5          0.8
```

A full table of our design can be found below:

n_estimators	max_depth	min_samples_split	min_samples_leaf	max_samples
50	1	2	1	0.5
50	1	2	1	0.8
50	1	2	5	0.5
50	1	2	5	0.8
50	1	6	1	0.5
50	1	6	1	0.8

n_estimators	max_depth	min_samples_split	min_samples_leaf	max_samples
50	1	6	5	0.5
50	1	6	5	0.8
50	20	2	1	0.5
50	20	2	1	0.8
50	20	2	5	0.5
50	20	2	5	0.8
50	20	6	1	0.5
50	20	6	1	0.8
50	20	6	5	0.5
50	20	6	5	0.8
100	1	2	1	0.5
100	1	2	1	0.8
100	1	2	5	0.5
100	1	2	5	0.8
100	1	6	1	0.5
100	1	6	1	0.8
100	1	6	5	0.5
100	1	6	5	0.8
100	20	2	1	0.5
100	20	2	1	0.8
100	20	2	5	0.5
100	20	2	5	0.8
100	20	6	1	0.5
100	20	6	1	0.8
100	20	6	5	0.5
100	20	6	5	0.8

Part 3: Modifying full factorial design to output a dictionary

I've modified the original `full_factorial_design` function and converted it to `generate_full_factorial_design`. The main changes are I now iterate through the items in design (a list itself) and create a small dictionary that has the keys and value pairs. I continue to do this until I have all combinations saved and return a list. The output is printed to give an illustration of what this looks like.

As an example use of our return (`design_matrix`), if we wanted the number of estimators for the first combination we could access this like `design_matrix[0]["n_estimators"]`

In [ ]:
```
# Standard imports
import os
import time

# Additional imports
import pandas as pd
import numpy as np
import itertools
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from itertools import combinations

# Local imports
# None

#---------------------------------------------------------------------
# Design matrix creation
#---------------------------------------------------------------------
# Define the factors and levels

# Provided values (yields accuracy 1.0 for all combos)
factors = {
    'n_estimators': [50, 100],
    'max_depth': [1, 20],
    'min_samples_split': [2, 6],
    'min_samples_leaf': [1, 5],
    'max_samples': [0.5, 0.8]
}

# New factors (yields accuracy 0.3 to 1.0)
# factors = {
#     'n_estimators': [5, 50],
#     'max_depth': [1, 10],
#     'min_samples_split': [2, 10],
#     'min_samples_leaf': [1, 10],
#     'max_samples': [0.1, 0.9]
# }

# Create a full factorial design
def generate_full_factorial_design(factors):
    levels = list(factors.values())
    design = list(itertools.product(*levels))
```

```
    # Convert each combination into a dictionary
    design_matrix = []
    for combo in design:
        single_design = {}
        for i, factor in enumerate(factors.keys()):
            single_design[factor] = combo[i]
        design_matrix.append(single_design)

    # Return list that has dicts inside
    return design_matrix

# Generate the design matrix
design_matrix = generate_full_factorial_design(factors)
print(design_matrix)
```

```
[{'n_estimators': 50, 'max_depth': 1, 'min_samples_split': 2, 'min_samples_leaf': 1,
'max_samples': 0.5}, {'n_estimators': 50, 'max_depth': 1, 'min_samples_split': 2, 'm
in_samples_leaf': 1, 'max_samples': 0.8}, {'n_estimators': 50, 'max_depth': 1, 'min_
samples_split': 2, 'min_samples_leaf': 5, 'max_samples': 0.5}, {'n_estimators': 50,
'max_depth': 1, 'min_samples_split': 2, 'min_samples_leaf': 5, 'max_samples': 0.8},
{'n_estimators': 50, 'max_depth': 1, 'min_samples_split': 6, 'min_samples_leaf': 1,
'max_samples': 0.5}, {'n_estimators': 50, 'max_depth': 1, 'min_samples_split': 6, 'm
in_samples_leaf': 1, 'max_samples': 0.8}, {'n_estimators': 50, 'max_depth': 1, 'min_
samples_split': 6, 'min_samples_leaf': 5, 'max_samples': 0.5}, {'n_estimators': 50,
'max_depth': 1, 'min_samples_split': 6, 'min_samples_leaf': 5, 'max_samples': 0.8},
{'n_estimators': 50, 'max_depth': 20, 'min_samples_split': 2, 'min_samples_leaf': 1,
'max_samples': 0.5}, {'n_estimators': 50, 'max_depth': 20, 'min_samples_split': 2,
'min_samples_leaf': 1, 'max_samples': 0.8}, {'n_estimators': 50, 'max_depth': 20, 'm
in_samples_split': 2, 'min_samples_leaf': 5, 'max_samples': 0.5}, {'n_estimators': 5
0, 'max_depth': 20, 'min_samples_split': 2, 'min_samples_leaf': 5, 'max_samples': 0.
8}, {'n_estimators': 50, 'max_depth': 20, 'min_samples_split': 6, 'min_samples_lea
f': 1, 'max_samples': 0.5}, {'n_estimators': 50, 'max_depth': 20, 'min_samples_spli
t': 6, 'min_samples_leaf': 1, 'max_samples': 0.8}, {'n_estimators': 50, 'max_depth':
20, 'min_samples_split': 6, 'min_samples_leaf': 5, 'max_samples': 0.5}, {'n_estimato
rs': 50, 'max_depth': 20, 'min_samples_split': 6, 'min_samples_leaf': 5, 'max_sample
s': 0.8}, {'n_estimators': 100, 'max_depth': 1, 'min_samples_split': 2, 'min_samples
_leaf': 1, 'max_samples': 0.5}, {'n_estimators': 100, 'max_depth': 1, 'min_samples_s
plit': 2, 'min_samples_leaf': 1, 'max_samples': 0.8}, {'n_estimators': 100, 'max_dep
th': 1, 'min_samples_split': 2, 'min_samples_leaf': 5, 'max_samples': 0.5}, {'n_esti
mators': 100, 'max_depth': 1, 'min_samples_split': 2, 'min_samples_leaf': 5, 'max_sa
mples': 0.8}, {'n_estimators': 100, 'max_depth': 1, 'min_samples_split': 6, 'min_sam
ples_leaf': 1, 'max_samples': 0.5}, {'n_estimators': 100, 'max_depth': 1, 'min_sampl
es_split': 6, 'min_samples_leaf': 1, 'max_samples': 0.8}, {'n_estimators': 100, 'max
_depth': 1, 'min_samples_split': 6, 'min_samples_leaf': 5, 'max_samples': 0.5}, {'n_
estimators': 100, 'max_depth': 1, 'min_samples_split': 6, 'min_samples_leaf': 5, 'ma
x_samples': 0.8}, {'n_estimators': 100, 'max_depth': 20, 'min_samples_split': 2, 'mi
n_samples_leaf': 1, 'max_samples': 0.5}, {'n_estimators': 100, 'max_depth': 20, 'min
_samples_split': 2, 'min_samples_leaf': 1, 'max_samples': 0.8}, {'n_estimators': 10
0, 'max_depth': 20, 'min_samples_split': 2, 'min_samples_leaf': 5, 'max_samples': 0.
5}, {'n_estimators': 100, 'max_depth': 20, 'min_samples_split': 2, 'min_samples_lea
f': 5, 'max_samples': 0.8}, {'n_estimators': 100, 'max_depth': 20, 'min_samples_spli
t': 6, 'min_samples_leaf': 1, 'max_samples': 0.5}, {'n_estimators': 100, 'max_dept
h': 20, 'min_samples_split': 6, 'min_samples_leaf': 1, 'max_samples': 0.8}, {'n_esti
mators': 100, 'max_depth': 20, 'min_samples_split': 6, 'min_samples_leaf': 5, 'max_s
amples': 0.5}, {'n_estimators': 100, 'max_depth': 20, 'min_samples_split': 6, 'min_s
amples_leaf': 5, 'max_samples': 0.8}]
```

Part 4: Updating the Random Forest Optimization Model

With some simple edits of adding our additional factors we can use the random forest optimization model to assess the different factors.

In [ ]:
```
#---------------------------------------------------------------------
# Run experiment
#---------------------------------------------------------------------
# Import the time library
t = Timer()
t.start()

# Load the Iris dataset
X, y = load_iris(return_X_y=True)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
results = []

# Run the experiments
for config in design_matrix:
    clf = RandomForestClassifier(n_estimators=config['n_estimators'], max_depth=con
    clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)

    # Store the results
    results.append({
        'n_estimators': config['n_estimators'],
        'max_depth': config['max_depth'],
        'min_samples_split': config['min_samples_split'],
        'min_samples_leaf': config['min_samples_leaf'],
        'max_samples': config['max_samples'],
        'accuracy': accuracy
    })

t.stop()  # A few seconds later

# Convert results to a DataFrame for analysis
results_df = pd.DataFrame(results)
print(results_df)
```

```
Elapsed time: 1.0591 seconds
Elapsed time: 0.0177 minutes
    n_estimators  max_depth  min_samples_split  min_samples_leaf  max_samples  \
0             50          1                  2                 1          0.5   
1             50          1                  2                 1          0.8   
2             50          1                  2                 5          0.5   
3             50          1                  2                 5          0.8   
4             50          1                  6                 1          0.5   
5             50          1                  6                 1          0.8   
6             50          1                  6                 5          0.5   
7             50          1                  6                 5          0.8   
8             50         20                  2                 1          0.5   
9             50         20                  2                 1          0.8   
10            50         20                  2                 5          0.5   
11            50         20                  2                 5          0.8   
12            50         20                  6                 1          0.5   
13            50         20                  6                 1          0.8   
14            50         20                  6                 5          0.5   
15            50         20                  6                 5          0.8   
16           100          1                  2                 1          0.5   
17           100          1                  2                 1          0.8   
18           100          1                  2                 5          0.5   
19           100          1                  2                 5          0.8   
20           100          1                  6                 1          0.5   
21           100          1                  6                 1          0.8   
22           100          1                  6                 5          0.5   
23           100          1                  6                 5          0.8   
24           100         20                  2                 1          0.5   
25           100         20                  2                 1          0.8   
26           100         20                  2                 5          0.5   
27           100         20                  2                 5          0.8   
28           100         20                  6                 1          0.5   
29           100         20                  6                 1          0.8   
30           100         20                  6                 5          0.5   
31           100         20                  6                 5          0.8   

    accuracy  
0        1.0  
1        1.0  
2        1.0  
3        1.0  
4        1.0  
5        1.0  
6        1.0  
7        1.0  
8        1.0  
9        1.0  
10       1.0  
11       1.0  
12       1.0  
13       1.0  
14       1.0  
15       1.0  
16       1.0  
17       1.0  
18       1.0  
```

```
19        1.0
20        1.0
21        1.0
22        1.0
23        1.0
24        1.0
25        1.0
26        1.0
27        1.0
28        1.0
29        1.0
30        1.0
31        1.0
```

A picture of our results are provided below. Notice that with default values the accuracy for all combos is 100%

Full Factorial Design and Percent Reacted:

	n_estimators	max_depth	min_samples_split	min_samples_leaf	max_samples	accuracy
0	5	1	2	1	0.1	0.900000
1	5	1	2	1	0.9	0.633333
2	5	1	2	10	0.1	0.333333
3	5	1	2	10	0.9	0.633333
4	5	1	10	1	0.1	0.900000
5	5	1	10	1	0.9	0.633333
6	5	1	10	10	0.1	0.333333
7	5	1	10	10	0.9	0.633333
8	5	10	2	1	0.1	0.966667
9	5	10	2	1	0.9	0.966667
10	5	10	2	10	0.1	0.333333
11	5	10	2	10	0.9	1.000000
12	5	10	10	1	0.1	0.900000
13	5	10	10	1	0.9	0.966667
14	5	10	10	10	0.1	0.333333
15	5	10	10	10	0.9	1.000000
16	50	1	2	1	0.1	0.966667
17	50	1	2	1	0.9	0.966667
18	50	1	2	10	0.1	0.366667
19	50	1	2	10	0.9	0.966667
20	50	1	10	1	0.1	0.966667
21	50	1	10	1	0.9	0.966667
22	50	1	10	10	0.1	0.366667
23	50	1	10	10	0.9	0.966667
24	50	10	2	1	0.1	1.000000
25	50	10	2	1	0.9	1.000000
26	50	10	2	10	0.1	0.366667
27	50	10	2	10	0.9	1.000000
28	50	10	10	1	0.1	0.966667
29	50	10	10	1	0.9	1.000000
30	50	10	10	10	0.1	0.366667
31	50	10	10	10	0.9	1.000000

Part 5: Analysis & Recommendations

To asssess the impact of different factors on the accuracy of combinations I did a few things:

- I adjusted the factors to those that were recommended in a follow up annoucnement, these new low and high values for each factor create more variance in accuracy to better see interactions and imapcts of effects
- I evaluted the impact of the main factors independently
- I evaluted the impact of different combinations of factors and calculated the contrast (the difference between the associated low level mean and high level mean which signifies the directional accuracy impact of a combination)

In [ ]:
```
"""
Purpose: Project 2 - Analysze methods for determining a 5 factor, 2 level model wit
Author: Syam Evani
"""

# Standard imports
import os
import time

# Additional imports
import pandas as pd
import numpy as np
import itertools
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from itertools import combinations

# Local imports
# None

#----------------------------------------------------------------------
# Timing examples
#----------------------------------------------------------------------
# Illustration of a Class Function for Error Exception
class TimerError(Exception):
    """A custom exception used to report errors in use of Timer class"""

## Illustration of a Class Function Measure Time Performance
class Timer:
    def __init__(self):
        self._start_time = None

    def start(self):
        """Start a new timer"""
        if self._start_time is not None:
            raise TimerError(f"Timer is running. Use .stop() to stop it")

        self._start_time = time.perf_counter()

    def stop(self):
        """Stop the timer, and report the elapsed time"""
        if self._start_time is None:
            raise TimerError(f"Timer is not running. Use .start() to start it")
```

```
        elapsed_time = time.perf_counter() - self._start_time
        elapsed_time_min = (time.perf_counter() - self._start_time)/60
        self._start_time = None
        print(f"Elapsed time: {elapsed_time:0.4f} seconds")
        print(f"Elapsed time: {elapsed_time_min:0.4f} minutes")

#---------------------------------------------------------------------
# Design matrix creation
#---------------------------------------------------------------------
# Define the factors and levels

# Provided values (yields accuracy 1.0 for all combos)
# factors = {
#     'n_estimators': [50, 100],
#     'max_depth': [1, 20],
#     'min_samples_split': [2, 6],
#     'min_samples_leaf': [1, 5],
#     'max_samples': [0.5, 0.8]
# }

# New factors (yields accuracy 0.3 to 1.0)
factors = {
    'n_estimators': [5, 50],
    'max_depth': [1, 10],
    'min_samples_split': [2, 10],
    'min_samples_leaf': [1, 10],
    'max_samples': [0.1, 0.9]
}

# Create a full factorial design
def generate_full_factorial_design(factors):
    levels = list(factors.values())
    design = list(itertools.product(*levels))

    # Convert each combination into a dictionary
    design_matrix = []
    for combo in design:
        single_design = {}
        for i, factor in enumerate(factors.keys()):
            single_design[factor] = combo[i]
        design_matrix.append(single_design)

    # Return list that has dicts inside
    return design_matrix

# Generate the design matrix
design_matrix = generate_full_factorial_design(factors)

#---------------------------------------------------------------------
# Run experiment
#---------------------------------------------------------------------
# Import the time library
t = Timer()
t.start()
```

```
# Load the Iris dataset
X, y = load_iris(return_X_y=True)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
results = []

# Run the experiments
for config in design_matrix:
    clf = RandomForestClassifier(n_estimators=config['n_estimators'], max_depth=con
    clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)

    # Store the results
    results.append({
        'n_estimators': config['n_estimators'],
        'max_depth': config['max_depth'],
        'min_samples_split': config['min_samples_split'],
        'min_samples_leaf': config['min_samples_leaf'],
        'max_samples': config['max_samples'],
        'accuracy': accuracy
    })

t.stop()  # A few seconds later

# Convert results to a DataFrame for analysis
results_df = pd.DataFrame(results)

#-------------------------------------------------------------------
# Assess the impact of main factors
#-------------------------------------------------------------------
main_effects = {}
for factor in factors.keys():
    main_effects[factor] = results_df.groupby(factor)['accuracy'].mean()

#-----------------------------------------------------------------------
# Assess the impact of interactions
#-----------------------------------------------------------------------
# Calculate interaction effects
interaction_effects = {}
factor_names = list(factors.keys())
for r in range(2, len(factor_names) + 1):
    for combo in combinations(factor_names, r):
        interaction_term = ' x '.join(combo)
        interaction_effects[interaction_term] = results_df.groupby(list(combo))['ac

#-----------------------------------------------------------------------
# Generate contrast Output
#-----------------------------------------------------------------------
contrast_output = pd.DataFrame(columns=['Factor/Interaction', 'Low Level Mean', 'Hi
contrast_rows = []

# Calculate contrast for main effects
for factor in factors.keys():
    low_level_mean = main_effects[factor].iloc[0]
```

```
    high_level_mean = main_effects[factor].iloc[1]
    effect = high_level_mean - low_level_mean
    contrast_rows.append({
        'Factor/Interaction': factor,
        'Low Level Mean': low_level_mean,
        'High Level Mean': high_level_mean,
        'Effect': effect
    })

# Calculate contrast for interaction effects
for interaction_term, interaction_data in interaction_effects.items():
    for idx, (level, row) in enumerate(interaction_data.iterrows()):
        low_level_mean = row.iloc[0]
        high_level_mean = row.iloc[1]
        effect = high_level_mean - low_level_mean
        contrast_rows.append({
            'Factor/Interaction': f'{interaction_term} (Level {level})',
            'Low Level Mean': low_level_mean,
            'High Level Mean': high_level_mean,
            'Effect': effect
        })

# Save contrast and sort from highest to lowest
contrast_output = pd.DataFrame(contrast_rows)
contrast_output = contrast_output.sort_values(by='Effect', ascending=False)

#-------------------------------------------------------------------------
# Print design, main effects contrast, and overall constrast rankings
#-------------------------------------------------------------------------
# Print design
print("Accuracy Report:")
print(results_df)

# Print main effects constrast
print("\nMain Effects:")
for factor, effects in main_effects.items():
    print(f"\n{factor}:")
    print(effects)

# Save contrast to text file for interactions
with open(os.path.join(os.getenv('USERPROFILE'),"repos","mece-6397-doe","project2",
    file.write(contrast_output.to_string(index=False))

# Print overall rankings
with open(os.path.join(os.getenv('USERPROFILE'),"repos","mece-6397-doe","project2",
    contents = file.read()
print(contents)
```

```
Elapsed time: 0.4008 seconds
Elapsed time: 0.0067 minutes

Main Effects:

n_estimators:
n_estimators
5     0.716667
50    0.827083
Name: accuracy, dtype: float64

max_depth:
max_depth
1     0.720833
10    0.822917
Name: accuracy, dtype: float64

min_samples_split:
min_samples_split
2     0.77500
10    0.76875
Name: accuracy, dtype: float64

min_samples_leaf:
min_samples_leaf
1     0.91875
10    0.62500
Name: accuracy, dtype: float64

max_samples:
max_samples
0.1    0.647917
0.9    0.895833
Name: accuracy, dtype: float64

Factor/Interaction  Low Level Mean  High Level Mean    Effect
                         n_estimators x max_depth x min_samples_leaf x max_samples
(Level (5, 10, 10))        0.333333         1.000000  0.666667
 n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Leve
l (5, 10, 10, 10))        0.333333         1.000000  0.666667
  n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Lev
el (5, 10, 2, 10))        0.333333         1.000000  0.666667
                                           max_depth x min_samples_leaf x max_sample
s (Level (10, 10))        0.350000         1.000000  0.650000
                    max_depth x min_samples_split x min_samples_leaf x max_samples (L
evel (10, 10, 10))        0.350000         1.000000  0.650000
                    max_depth x min_samples_split x min_samples_leaf x max_samples
(Level (10, 2, 10))        0.350000         1.000000  0.650000
 n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Leve
l (50, 10, 2, 10))        0.366667         1.000000  0.633333
                        n_estimators x max_depth x min_samples_leaf x max_samples (L
evel (50, 10, 10))        0.366667         1.000000  0.633333
n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Level
(50, 10, 10, 10))        0.366667         1.000000  0.633333
                                        n_estimators x min_samples_leaf x max_sample
s (Level (50, 10))        0.366667         0.983333  0.616667
```

```
                n_estimators x min_samples_split x min_samples_leaf x max_samples (L
evel (50, 10, 10))        0.366667         0.983333  0.616667
                n_estimators x min_samples_split x min_samples_leaf x max_samples
(Level (50, 2, 10))        0.366667         0.983333  0.616667
                        n_estimators x max_depth x min_samples_leaf x max_samples
(Level (50, 1, 10))        0.366667         0.966667  0.600000
 n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Leve
l (50, 1, 10, 10))        0.366667         0.966667  0.600000
  n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Lev
el (50, 1, 2, 10))        0.366667         0.966667  0.600000
                                  min_samples_split x min_samples_leaf x max_sample
s (Level (10, 10))        0.350000         0.900000  0.550000
                                   min_samples_split x min_samples_leaf x max_sampl
es (Level (2, 10))        0.350000         0.900000  0.550000
                                                            min_samples_leaf x max_
samples (Level 10)        0.350000         0.900000  0.550000
                  n_estimators x min_samples_split x min_samples_leaf x max_samples
(Level (5, 2, 10))        0.333333         0.816667  0.483333
                n_estimators x min_samples_split x min_samples_leaf x max_samples
(Level (5, 10, 10))        0.333333         0.816667  0.483333
                                        n_estimators x min_samples_leaf x max_sampl
es (Level (5, 10))        0.333333         0.816667  0.483333
                   max_depth x min_samples_split x min_samples_leaf x max_samples
(Level (1, 10, 10))        0.350000         0.800000  0.450000
                     max_depth x min_samples_split x min_samples_leaf x max_samples
(Level (1, 2, 10))        0.350000         0.800000  0.450000
                                           max_depth x min_samples_leaf x max_sampl
es (Level (1, 10))        0.350000         0.800000  0.450000
                        n_estimators x max_depth x min_samples_split x max_samples
(Level (5, 10, 10))        0.616667         0.983333  0.366667
                                               n_estimators x max_depth x max_sampl
es (Level (5, 10))        0.633333         0.983333  0.350000
                                         max_depth x min_samples_split x max_sample
s (Level (10, 10))        0.641667         0.991667  0.350000
                                                                   max_depth x max_
samples (Level 10)        0.654167         0.991667  0.337500
                         n_estimators x max_depth x min_samples_split x max_samples
(Level (5, 10, 2))        0.650000         0.983333  0.333333
                       n_estimators x max_depth x min_samples_split x max_samples (L
evel (50, 10, 10))        0.666667         1.000000  0.333333
                                              n_estimators x max_depth x max_sample
s (Level (50, 10))        0.675000         1.000000  0.325000
                                          max_depth x min_samples_split x max_sampl
es (Level (10, 2))        0.666667         0.991667  0.325000
                                      n_estimators x min_samples_split x max_sample
s (Level (50, 10))        0.666667         0.983333  0.316667
                        n_estimators x max_depth x min_samples_split x max_samples
(Level (50, 10, 2))        0.683333         1.000000  0.316667
                                                                n_estimators x max_
samples (Level 50)        0.670833         0.983333  0.312500
                                       n_estimators x min_samples_split x max_sampl
es (Level (50, 2))        0.675000         0.983333  0.308333
                        n_estimators x max_depth x min_samples_split x max_samples
(Level (50, 1, 10))        0.666667         0.966667  0.300000
                                               n_estimators x max_depth x max_sampl
es (Level (50, 1))        0.666667         0.966667  0.300000
```

```
                         n_estimators x max_depth x min_samples_split x max_samples
(Level (50, 1, 2))        0.666667         0.966667  0.300000
   n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Le
vel (5, 1, 2, 10))        0.333333         0.633333  0.300000
  n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Lev
el (5, 1, 10, 10))        0.333333         0.633333  0.300000
                          n_estimators x max_depth x min_samples_leaf x max_samples
(Level (5, 1, 10))        0.333333         0.633333  0.300000
                                                            min_samples_split x max_
samples (Level 10)        0.641667         0.895833  0.254167

max_samples        0.647917         0.895833  0.247917
                                                             min_samples_split x max
_samples (Level 2)        0.654167         0.895833  0.241667
                                        n_estimators x min_samples_split x max_sampl
es (Level (5, 10))        0.616667         0.808333  0.191667
                                                                    n_estimators x m
ax_depth (Level 5)        0.625000         0.808333  0.183333
                                                                  n_estimators x max
_samples (Level 5)        0.625000         0.808333  0.183333
                                         n_estimators x min_samples_split x max_samp
les (Level (5, 2))        0.633333         0.808333  0.175000
                                                                     max_depth x max
_samples (Level 1)        0.641667         0.800000  0.158333
                                           max_depth x min_samples_split x max_sampl
es (Level (1, 10))        0.641667         0.800000  0.158333
                                            max_depth x min_samples_split x max_samp
les (Level (1, 2))        0.641667         0.800000  0.158333

n_estimators        0.716667         0.827083  0.110417

max_depth        0.720833         0.822917  0.102083
  n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Lev
el (5, 10, 10, 1))        0.900000         0.966667  0.066667
                    max_depth x min_samples_split x min_samples_leaf x max_samples
(Level (10, 10, 1))        0.933333         0.983333  0.050000
 n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Leve
l (50, 10, 10, 1))        0.966667         1.000000  0.033333
                          n_estimators x max_depth x min_samples_leaf x max_samples
(Level (5, 10, 1))        0.933333         0.966667  0.033333
                                            max_depth x min_samples_leaf x max_sampl
es (Level (10, 1))        0.958333         0.983333  0.025000
                                                                   n_estimators x ma
x_depth (Level 50)        0.816667         0.837500  0.020833
                 n_estimators x min_samples_split x min_samples_leaf x max_samples
(Level (50, 10, 1))        0.966667         0.983333  0.016667
                                                 n_estimators x max_depth x max_samp
les (Level (5, 1))        0.616667         0.633333  0.016667
                         n_estimators x max_depth x min_samples_leaf x max_samples
(Level (50, 10, 1))        0.983333         1.000000  0.016667
                          n_estimators x max_depth x min_samples_split x max_samples
(Level (5, 1, 2))        0.616667         0.633333  0.016667
                         n_estimators x max_depth x min_samples_split x max_samples
(Level (5, 1, 10))        0.616667         0.633333  0.016667
                                         n_estimators x min_samples_leaf x max_sampl
es (Level (50, 1))        0.975000         0.983333  0.008333
```

```
  n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Lev
el (50, 10, 2, 1))        1.000000         1.000000  0.000000
                                         n_estimators x max_depth x min_samples_sp
lit (Level (5, 1))        0.625000         0.625000  0.000000
                     max_depth x min_samples_split x min_samples_leaf x max_samples
(Level (10, 2, 1))        0.983333         0.983333  0.000000
                                                               max_depth x min_sampl
es_split (Level 1)        0.720833         0.720833  0.000000
                  n_estimators x min_samples_split x min_samples_leaf x max_samples
(Level (50, 2, 1))        0.983333         0.983333  0.000000
                          n_estimators x max_depth x min_samples_leaf x max_samples
(Level (50, 1, 1))        0.966667         0.966667  0.000000
   n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Le
vel (50, 1, 2, 1))        0.966667         0.966667  0.000000
   n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Le
vel (5, 10, 2, 1))        0.966667         0.966667  0.000000
  n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Lev
el (50, 1, 10, 1))        0.966667         0.966667  0.000000
                                        n_estimators x max_depth x min_samples_spl
it (Level (50, 1))        0.816667         0.816667  0.000000
                                                           n_estimators x min_sample
s_split (Level 50)        0.829167         0.825000 -0.004167

min_samples_split        0.775000         0.768750 -0.006250
                                                            n_estimators x min_sampl
es_split (Level 5)        0.720833         0.712500 -0.008333
                                       n_estimators x max_depth x min_samples_spli
t (Level (50, 10))        0.841667         0.833333 -0.008333
                                                              max_depth x min_sample
s_split (Level 10)        0.829167         0.816667 -0.012500
                                        n_estimators x max_depth x min_samples_spl
it (Level (5, 10))        0.816667         0.800000 -0.016667
                                   min_samples_split x min_samples_leaf x max_sampl
es (Level (10, 1))        0.933333         0.891667 -0.041667
                                                              min_samples_leaf x max
_samples (Level 1)        0.945833         0.891667 -0.054167
                                    min_samples_split x min_samples_leaf x max_samp
les (Level (2, 1))        0.958333         0.891667 -0.066667
                  n_estimators x min_samples_split x min_samples_leaf x max_samples
(Level (5, 10, 1))        0.900000         0.800000 -0.100000
                                        n_estimators x min_samples_leaf x max_samp
les (Level (5, 1))        0.916667         0.800000 -0.116667
                                           max_depth x min_samples_leaf x max_samp
les (Level (1, 1))        0.933333         0.800000 -0.133333
                   n_estimators x min_samples_split x min_samples_leaf x max_samples
(Level (5, 2, 1))        0.933333         0.800000 -0.133333
                      max_depth x min_samples_split x min_samples_leaf x max_samples
(Level (1, 2, 1))        0.933333         0.800000 -0.133333
                     max_depth x min_samples_split x min_samples_leaf x max_samples
(Level (1, 10, 1))        0.933333         0.800000 -0.133333
                           n_estimators x max_depth x min_samples_leaf x max_samples
(Level (5, 1, 1))        0.900000         0.633333 -0.266667
   n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Le
vel (5, 1, 10, 1))        0.900000         0.633333 -0.266667
                   n_estimators x max_depth x min_samples_split x min_samples_leaf
(Level (5, 10, 10))        0.933333         0.666667 -0.266667
```

```
    n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (L
evel (5, 1, 2, 1))        0.900000         0.633333 -0.266667
                                   n_estimators x min_samples_split x min_samples_le
af (Level (5, 10))        0.850000         0.575000 -0.275000
                     n_estimators x max_depth x min_samples_split x min_samples_leaf
(Level (5, 1, 2))        0.766667         0.483333 -0.283333
                    n_estimators x max_depth x min_samples_split x min_samples_leaf
(Level (5, 1, 10))        0.766667         0.483333 -0.283333
                                     max_depth x min_samples_split x min_samples_lea
f (Level (10, 10))        0.958333         0.675000 -0.283333
                                            n_estimators x max_depth x min_samples_l
eaf (Level (5, 1))        0.766667         0.483333 -0.283333
                                           n_estimators x max_depth x min_samples_le
af (Level (5, 10))        0.950000         0.666667 -0.283333
                                                             n_estimators x min_samp
les_leaf (Level 5)        0.858333         0.575000 -0.283333
                                                       min_samples_split x min_sampl
es_leaf (Level 10)        0.912500         0.625000 -0.287500
                                                                max_depth x min_samp
les_leaf (Level 1)        0.866667         0.575000 -0.291667
                                      max_depth x min_samples_split x min_samples_le
af (Level (1, 10))        0.866667         0.575000 -0.291667
                                       max_depth x min_samples_split x min_samples_l
eaf (Level (1, 2))        0.866667         0.575000 -0.291667
                                    n_estimators x min_samples_split x min_samples_l
eaf (Level (5, 2))        0.866667         0.575000 -0.291667

min_samples_leaf        0.918750         0.625000 -0.293750
                                                               max_depth x min_sampl
es_leaf (Level 10)        0.970833         0.675000 -0.295833
                                  n_estimators x min_samples_split x min_samples_lea
f (Level (50, 10))        0.975000         0.675000 -0.300000
                                           n_estimators x max_depth x min_samples_le
af (Level (50, 1))        0.966667         0.666667 -0.300000
                                                        min_samples_split x min_samp
les_leaf (Level 2)        0.925000         0.625000 -0.300000
                  n_estimators x max_depth x min_samples_split x min_samples_leaf (L
evel (50, 10, 10))        0.983333         0.683333 -0.300000
                    n_estimators x max_depth x min_samples_split x min_samples_leaf
(Level (5, 10, 2))        0.966667         0.666667 -0.300000
                    n_estimators x max_depth x min_samples_split x min_samples_leaf
(Level (50, 1, 2))        0.966667         0.666667 -0.300000
                   n_estimators x max_depth x min_samples_split x min_samples_leaf
(Level (50, 1, 10))        0.966667         0.666667 -0.300000
                                                            n_estimators x min_sampl
es_leaf (Level 50)        0.979167         0.675000 -0.304167
                                          n_estimators x max_depth x min_samples_lea
f (Level (50, 10))        0.991667         0.683333 -0.308333
                                      max_depth x min_samples_split x min_samples_le
af (Level (10, 2))        0.983333         0.675000 -0.308333
                                   n_estimators x min_samples_split x min_samples_le
af (Level (50, 2))        0.983333         0.675000 -0.308333
                   n_estimators x max_depth x min_samples_split x min_samples_leaf
(Level (50, 10, 2))        1.000000         0.683333 -0.316667
```

Firstly, our new accuracy report with these updated factor value yields the image below. Note the factors that yield the highest accuracy (1.0) are the following and are combos I would suggest for best accuracy:

- number of estimators: 5, max depth: 10, minimum samples split: 2, minimum samples leaf: 10, maximum samples: 0.9
- number of estimators: 5, max depth: 10, minimum samples split: 10, minimum samples leaf: 10, maximum samples: 0.9
- number of estimators: 50, max depth: 10, minimum samples split: 2, minimum samples leaf: 10, maximum samples: 0.1
- number of estimators: 50, max depth: 10, minimum samples split: 2, minimum samples leaf: 10, maximum samples: 0.9
- number of estimators: 50, max depth: 10, minimum samples split: 10, minimum samples leaf: 1, maximum samples: 0.9
- number of estimators: 50, max depth: 10, minimum samples split: 10, minimum samples leaf: 10, maximum samples: 0.9

```
Accuracy Report:
    n_estimators  max_depth  min_samples_split  min_samples_leaf  max_samples  accuracy
0              5          1                  2                 1          0.1  0.900000
1              5          1                  2                 1          0.9  0.633333
2              5          1                  2                10          0.1  0.333333
3              5          1                  2                10          0.9  0.633333
4              5          1                 10                 1          0.1  0.900000
5              5          1                 10                 1          0.9  0.633333
6              5          1                 10                10          0.1  0.333333
7              5          1                 10                10          0.9  0.633333
8              5         10                  2                 1          0.1  0.966667
9              5         10                  2                 1          0.9  0.966667
10             5         10                  2                10          0.1  0.333333
11             5         10                  2                10          0.9  1.000000
12             5         10                 10                 1          0.1  0.900000
13             5         10                 10                 1          0.9  0.966667
14             5         10                 10                10          0.1  0.333333
15             5         10                 10                10          0.9  1.000000
16            50          1                  2                 1          0.1  0.966667
17            50          1                  2                 1          0.9  0.966667
18            50          1                  2                10          0.1  0.366667
19            50          1                  2                10          0.9  0.966667
20            50          1                 10                 1          0.1  0.966667
21            50          1                 10                 1          0.9  0.966667
22            50          1                 10                10          0.1  0.366667
23            50          1                 10                10          0.9  0.966667
24            50         10                  2                 1          0.1  1.000000
25            50         10                  2                 1          0.9  1.000000
26            50         10                  2                10          0.1  0.366667
27            50         10                  2                10          0.9  1.000000
28            50         10                 10                 1          0.1  0.966667
29            50         10                 10                 1          0.9  1.000000
30            50         10                 10                10          0.1  0.366667
31            50         10                 10                10          0.9  1.000000
```

The complete interaction and contrast report is provided below. Perhaps the most clear takeway is increasing maximum samples has a strong impact on increasing accuracy.

Following that, several complex relationships exist, most notably those combos of factors that combine to form a perfect accuracy.

Factor/Interaction	Low Level Mean	High Level Mean	Effect
n_estimators x max_depth x min_samples_leaf x max_samples (Level (5, 10, 10))	0.333333	1.000000	0.666667
n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Level (5, 10, 10, 10))	0.333333	1.000000	0.666667
n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Level (5, 10, 2, 10))	0.333333	1.000000	0.666667
max_depth x min_samples_leaf x max_samples (Level (10, 10))	0.350000	1.000000	0.650000
max_depth x min_samples_split x min_samples_leaf x max_samples (Level (10, 10, 10))	0.350000	1.000000	0.650000
max_depth x min_samples_split x min_samples_leaf x max_samples (Level (10, 2, 10))	0.350000	1.000000	0.650000
n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Level (50, 10, 2, 10))	0.366667	1.000000	0.633333
n_estimators x max_depth x min_samples_leaf x max_samples (Level (50, 10, 10))	0.366667	1.000000	0.633333
n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Level (50, 10, 10, 10))	0.366667	1.000000	0.633333
n_estimators x min_samples_leaf x max_samples (Level (50, 10))	0.366667	0.983333	0.616667
n_estimators x min_samples_split x min_samples_leaf x max_samples (Level (50, 10, 10))	0.366667	0.983333	0.616667
n_estimators x min_samples_split x min_samples_leaf x max_samples (Level (50, 2, 10))	0.366667	0.983333	0.616667
n_estimators x max_depth x min_samples_leaf x max_samples (Level (50, 1, 10))	0.366667	0.966667	0.600000
n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Level (50, 1, 10, 10))	0.366667	0.966667	0.600000
n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Level (50, 1, 2, 10))	0.366667	0.966667	0.600000
min_samples_split x min_samples_leaf x max_samples (Level (10, 10))	0.350000	0.900000	0.550000
min_samples_split x min_samples_leaf x max_samples (Level (2, 10))	0.350000	0.900000	0.550000
min_samples_leaf x max_samples (Level 10)	0.350000	0.900000	0.550000

Factor/Interaction	Low Level Mean	High Level Mean	Effect
n_estimators x min_samples_split x min_samples_leaf x max_samples (Level (5, 2, 10))	0.333333	0.816667	0.483333
n_estimators x min_samples_split x min_samples_leaf x max_samples (Level (5, 10, 10))	0.333333	0.816667	0.483333
n_estimators x min_samples_leaf x max_samples (Level (5, 10))	0.333333	0.816667	0.483333
max_depth x min_samples_split x min_samples_leaf x max_samples (Level (1, 10, 10))	0.350000	0.800000	0.450000
max_depth x min_samples_split x min_samples_leaf x max_samples (Level (1, 2, 10))	0.350000	0.800000	0.450000
max_depth x min_samples_leaf x max_samples (Level (1, 10))	0.350000	0.800000	0.450000
n_estimators x max_depth x min_samples_split x max_samples (Level (5, 10, 10))	0.616667	0.983333	0.366667
n_estimators x max_depth x max_samples (Level (5, 10))	0.633333	0.983333	0.350000
max_depth x min_samples_split x max_samples (Level (10, 10))	0.641667	0.991667	0.350000
max_depth x max_samples (Level 10)	0.654167	0.991667	0.337500
n_estimators x max_depth x min_samples_split x max_samples (Level (5, 10, 2))	0.650000	0.983333	0.333333
n_estimators x max_depth x min_samples_split x max_samples (Level (50, 10, 10))	0.666667	1.000000	0.333333
n_estimators x max_depth x max_samples (Level (50, 10))	0.675000	1.000000	0.325000
max_depth x min_samples_split x max_samples (Level (10, 2))	0.666667	0.991667	0.325000
n_estimators x min_samples_split x max_samples (Level (50, 10))	0.666667	0.983333	0.316667
n_estimators x max_depth x min_samples_split x max_samples (Level (50, 10, 2))	0.683333	1.000000	0.316667
n_estimators x max_samples (Level 50)	0.670833	0.983333	0.312500
n_estimators x min_samples_split x max_samples (Level (50, 2))	0.675000	0.983333	0.308333
n_estimators x max_depth x min_samples_split x max_samples (Level (50, 1, 10))	0.666667	0.966667	0.300000
n_estimators x max_depth x max_samples (Level (50, 1))	0.666667	0.966667	0.300000
n_estimators x max_depth x min_samples_split x	0.666667	0.966667	0.300000

Factor/Interaction	Low Level Mean	High Level Mean	Effect
max_samples (Level (50, 1, 2))			
n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Level (5, 1, 2, 10))	0.333333	0.633333	0.300000
n_estimators x max_depth x min_samples_split x min_samples_leaf x max_samples (Level (5, 1, 10, 10))	0.333333	0.633333	0.300000
n_estimators x max_depth x min_samples_leaf x max_samples (Level (5, 1, 10))	0.333333	0.633333	0.300000
min_samples_split x max_samples (Level 10)	0.641667	0.895833	0.254167
max_samples	0.647917	0.895833	0.247917
min_samples_split x max_samples (Level 2)	0.654167	0.895833	0.241667
n_estimators x min_samples_split x max_samples (Level (5, 10))	0.616667	0.808333	0.191667
n_estimators x min_samples_split x max_samples (Level (5, 2))	0.616667	0.808333	0.191667
max_depth x min_samples_split x max_samples (Level (1, 10))	0.650000	0.825000	0.175000
max_depth x max_samples (Level 1)	0.654167	0.825000	0.170833
max_depth x min_samples_split x max_samples (Level (1, 2))	0.666667	0.825000	0.158333
min_samples_split x max_samples (Level 1)	0.670833	0.820833	0.150000
max_depth (Level 1)	0.658333	0.791667	0.133333
n_estimators (Level 5)	0.633333	0.750000	0.116667
n_estimators x max_depth (Level 50)	0.670833	0.766667	0.095833
n_estimators (Level 50)	0.670833	0.750000	0.079167
max_depth (Level 10)	0.658333	0.716667	0.058333
n_estimators x max_depth (Level 1)	0.666667	0.716667	0.050000
min_samples_split (Level 1)	0.662500	0.700000	0.037500
n_estimators (Level 1)	0.670833	0.691667	0.020833
min_samples_split (Level 10)	0.658333	0.675000	0.016667
min_samples_split (Level 2)	0.662500	0.675000	0.012500