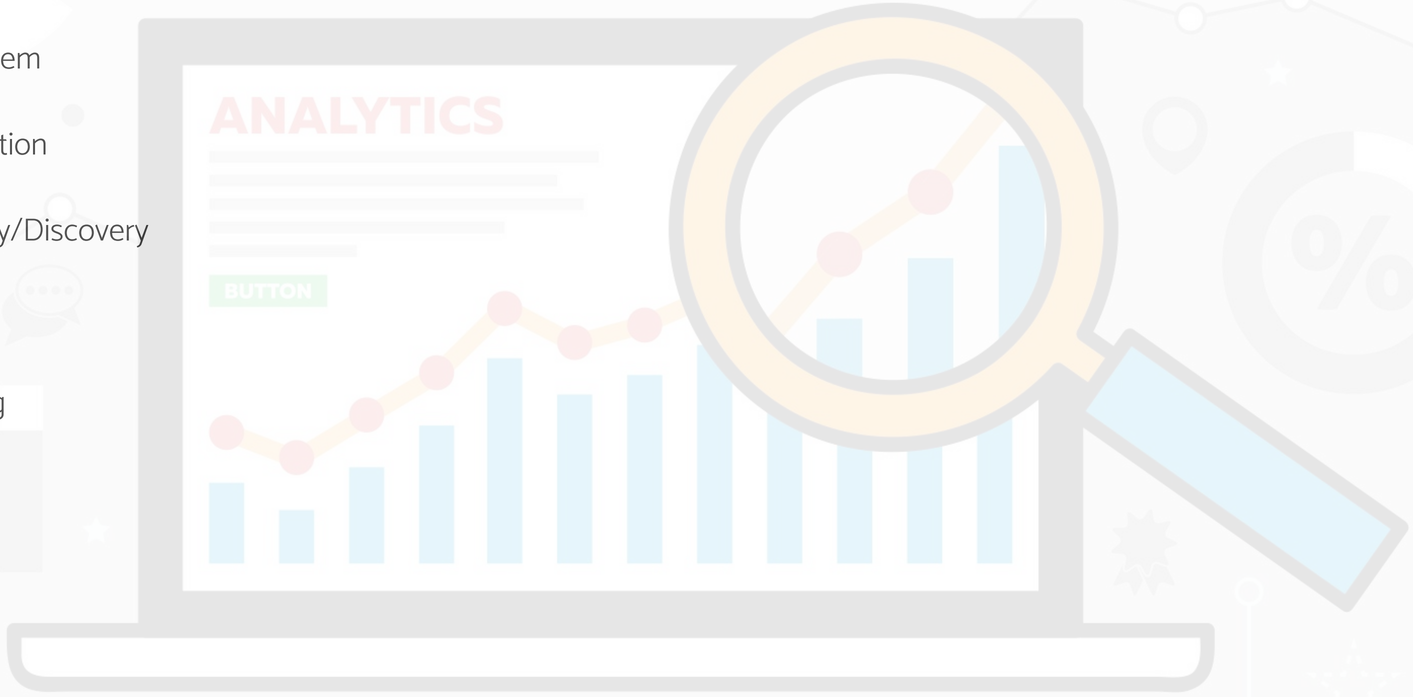
Predicting Productivity

Kevin Chu, Gregory McKernan, Sero Nazarian

Agenda

1. Business Problem
2. Proposed Solution
3. Data Dictionary/Discovery
4. Feature Prep
5. Model Running
6. Results
7. Next Steps



What is employee productivity

- Simply, it is defined as the output of work produced by an employee at a certain point in time.

- Measurement is an important tool to help
  - Retain high performing talent
  - Let go of underperforming resources
  - Understanding of work conditions and how to get the best out of employees
  - Make sure from the work being done, quality of work is sustained

Current & Future state of employee productivity

- Historically employee productivity could be measured on a quantitative and qualitative level since managers were physically able to see their employees

- Due to the pandemic on average almost 70% of the workforce is working remotely
    - An average increase of 47% of productivity as a result

Business Problem

In a labour intensive factory environment with many manual processes and overtime can we predict worker productivity?

Using predictive analysis, we can try to predict worker productivity so that plant owners can make better decisions about how much they can produce, how long it will take and how much it will cost.

Dataset

Fast Fashion has created a increased demand for clothing and garments.

Turning raw materials into clothes is labor intensive.

Our dataset was pulled from the UCI database and is an export of factory data and the actual productivity of workers.

Data Exploration

In [3]:
```
data.head()
```

Out[3]:

	date	quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_wo
0	1/1/2015	Quarter1	sweing	Thursday	8	0.80	26.16	1108.0	7080	98	0.0	0	0	
1	1/1/2015	Quarter1	finishing	Thursday	1	0.75	3.94	NaN	960	0	0.0	0	0	
2	1/1/2015	Quarter1	sweing	Thursday	11	0.80	11.41	968.0	3660	50	0.0	0	0	
3	1/1/2015	Quarter1	sweing	Thursday	12	0.80	11.41	968.0	3660	50	0.0	0	0	
4	1/1/2015	Quarter1	sweing	Thursday	6	0.80	25.90	1170.0	1920	50	0.0	0	0	

Data Dictionary

01 date : Date in MM-DD-YYYY
02 day : Day of the Week
03 quarter : A portion of the month. A month was divided into four quarters
04 department : Associated department with the instance
05 team_no : Associated team number with the instance
06 no_of_workers : Number of workers in each team
07 no_of_style_change : Number of changes in the style of a particular product
08 targeted_productivity : Targeted productivity set by the Authority for each team for each day.
09 smv : Standard Minute Value, it is the allocated time for a task
10 wip : Work in progress. Includes the number of unfinished items for products
11 over_time : Represents the amount of overtime by each team in minutes
12 incentive : Represents the amount of financial incentive (in BDT) that enables or motivates a particular course of action.
13 idle_time : The amount of time when the production was interrupted due to several reasons
14 idle_men : The number of workers who were idle due to production interruption
15 actual_productivity : The actual % of productivity that was delivered by the workers. It ranges from 0-1.

Feature Prep

- To get the data ready for processing we identified missing values and naming conventions that might throw errors in our results

- Converted objects to binary for all int64 dataset

```
In [10]: data.department.value_counts()

Out[10]: sweing        691
         finishing     257
         finishing     249
         Name: department, dtype: int64

In [11]: data.loc[:,'department'] = data.loc[:,'department'].str.strip()

In [12]: data.department.value_counts()

Out[12]: sweing       691
         finishing    506
         Name: department, dtype: int64
```

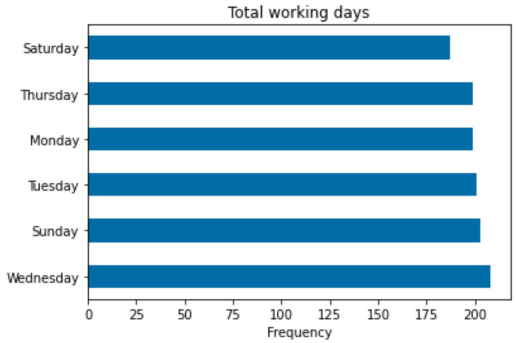
```
In [4]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1197 entries, 0 to 1196
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   date                   1197 non-null   object
 1   quarter                1197 non-null   object
 2   department             1197 non-null   object
 3   day                    1197 non-null   object
 4   team                   1197 non-null   int64
 5   targeted_productivity  1197 non-null   float64
 6   smv                    1197 non-null   float64
 7   wip                    691 non-null    float64
 8   over_time              1197 non-null   int64
 9   incentive              1197 non-null   int64
 10  idle_time              1197 non-null   float64
 11  idle_men               1197 non-null   int64
 12  no_of_style_change     1197 non-null   int64
 13  no_of_workers          1197 non-null   float64
 14  actual_productivity    1197 non-null   float64
dtypes: float64(6), int64(5), object(4)
memory usage: 140.4+ KB
```

```
In [7]: ##Fill in work in progress values
        data.isnull().sum()

Out[7]: date                       0
        quarter                    0
        department                 0
        day                        0
        team                       0
        targeted_productivity      0
        smv                        0
        wip                      506
        over_time                  0
        incentive                  0
        idle_time                  0
        idle_men                   0
        no_of_style_change         0
        no_of_workers              0
        actual_productivity        0
        dtype: int64
```

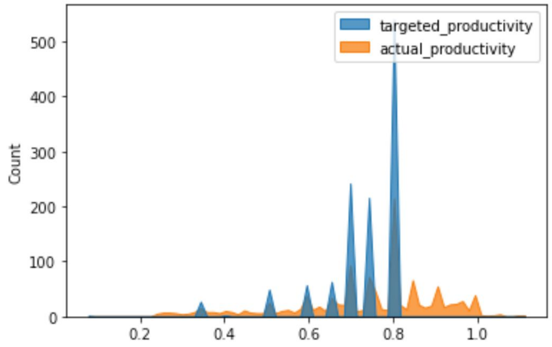
Data Discovery

- Data Visualization - Use of Bar charts and Histograms to visualize data

In [13]:
```
data['day'].value_counts().plot(kind='barh')
plt.title("Total working days")
plt.xlabel('Frequency')
plt.show()
```




```
sns.histplot(data=data[['targeted_productivity', 'actual_productivity']], element='poly')
plt.show()
```

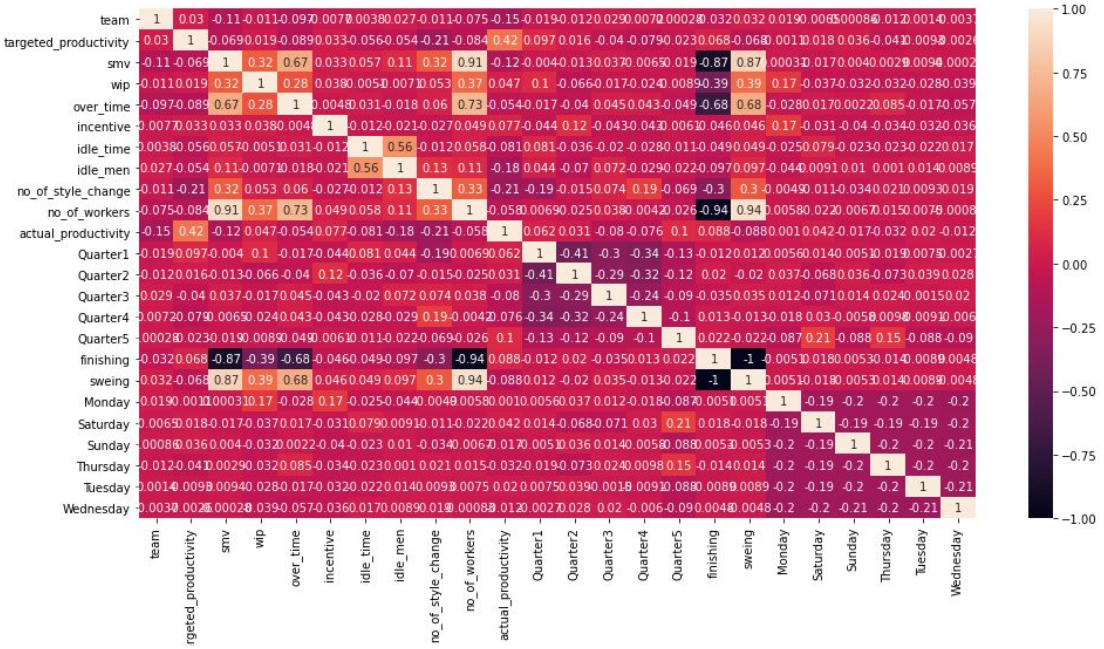


Data Discovery Cont.

- Used a correlation heatmap to identify meaningful relationships and decide on what variables to use/drop
- In this scenario we'd expect incentive to highly correlate with productivity (target or actual) but it is not the case

In [20]:
```
plt.figure(figsize=(16,8))
sns.heatmap(data.corr(),annot=True)
```

Out[20]: <AxesSubplot:>



Supervised Learning

The ability to predict the amount of garments that would be produced will use regression analysis, this is a supervised learning process as the data is labeled and we are working to find one output based on multiple inputs.

Modeling Methods

Linear Regression

A linear model will use the coefficients specified to minimize the residual sum of the squares between the observed data and the predicted values of the linear approximation.

Random Forest Regression

The random forest algorithm proposed, by Breiman in 2001, has been extremely successful as a general-purpose classification and regression method .
RF is one of the most powerful ensemble methods with high performance when dealing with high dimensional data.

Decision Tree Regressor

Build a decision tree regressor from the training set (X, y).

Decision tree is used to fit a sine curve with addition noisy observation. As a result, it learns local linear regressions approximating the sine curve.

Results with all features

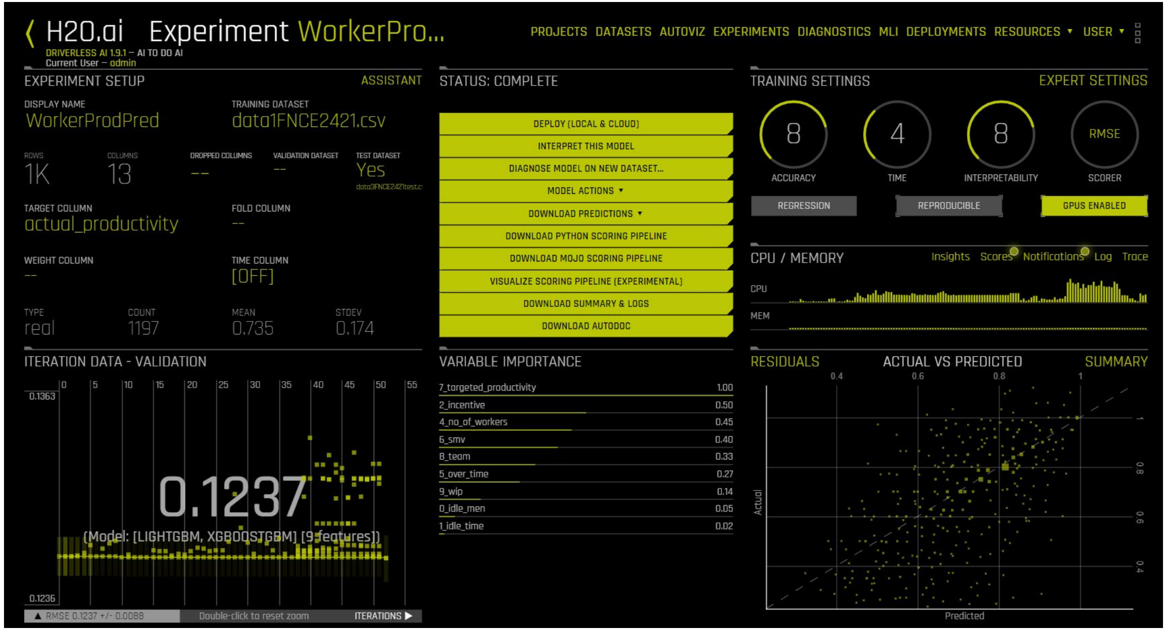
```
In [27]: clf = LinearRegression()
         clf.fit(x_train2, y_train)
         y_pred_linear=clf.predict(x_test2)
         acc_linear=round( clf.score(x_train2, y_train) * 100, 2)
         print ('score:'+str(acc_linear) + ' percent')

         score:28.39 percent

In [28]: clf = RandomForestRegressor(n_estimators=100)
         clf.fit(x_train2, y_train)
         y_pred_rf=clf.predict(x_test2)
         acc_rf= round(clf.score(x_train2, y_train) * 100, 2)
         print ("Accuracy: %i %% \n"%acc_rf)

         Accuracy: 92 %

In [29]: clf=DecisionTreeRegressor()
         clf.fit(x_train2, y_train)
         y_pred_dt= clf.predict(x_test2)
         acc_dt = round( clf.score(x_train2, y_train) * 100, 2)
         print (str(acc_dt) + ' percent')

         100.0 percent
```

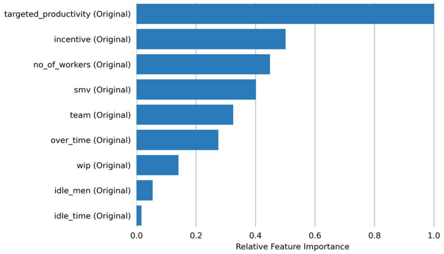
Results with select features

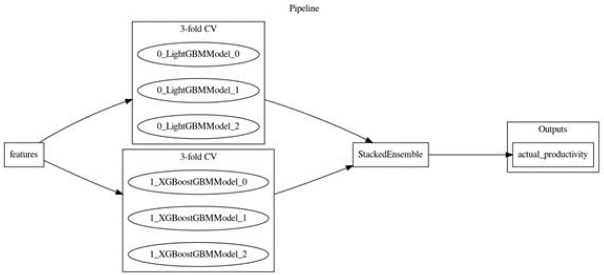
```
In [37]: clf = LinearRegression()
         clf.fit(x_train2, y_train)
         y_pred_linear=clf.predict(x_test2)
         acc_linear=round( clf.score(x_train2, y_train) * 100, 2)
         print ('score:'+str(acc_linear) + ' percent')

         score:27.28 percent

In [38]: clf = RandomForestRegressor(n_estimators=100)
         clf.fit(x_train2, y_train)
         y_pred_rf=clf.predict(x_test2)
         acc_rf= round(clf.score(x_train2, y_train) * 100, 2)
         print ("Accuracy: %i %% \n"%acc_rf)

         Accuracy: 83 %

In [39]: clf=DecisionTreeRegressor()
         clf.fit(x_train2, y_train)
         y_pred_dt= clf.predict(x_test2)
         acc_dt = round( clf.score(x_train2, y_train) * 100, 2)
         print (str(acc_dt) + ' percent')

         87.64 percent
```

- Used only the first 11 columns of int64 data and excluded the categorical variables that we used ‘get_dummies’ on.

Working with Auto ML







Scorer	Final ensemble scores on validation (internal or external holdout(s)) data
RMSE	0.1237113

Real World Application

- We realized these models and accuracy would be exclusive to this dataset.

- In our data we did not encounter any risk of outliers, or need intensive data cleansing

- However, companies can leverage their own data libraries to select features exclusive to the industry, work environment, and other outside factors to apply to the same regression models we used in this exercise

- With the combination of use of code in python and applications like Auto ML the finance industry will be able to make some strong advancements in automation, forecasting, and reducing redundant tasks

Thank you!