20/06/2020

Employee Absenteeism

Submitted by:

Aniket Patil

Raj Ahuja

Yashraj Khokrale

Prathamesh Shelar

Simple > Complex

Features > Algorithms

Terminology

The following terms will come up repeatedly in our discussion of effective machine learning:

- **Instance**: The thing about which you want to make a prediction. For example, the instance might be a web page that you want to classify as either "about cats" or "not about cats".

- **Label**: An answer for a prediction task either the answer produced by a machine learning system, or the right answer supplied in training data. For example, the label for a web page might be "about cats".

- **Feature**: A property of an instance used in a prediction task. For example, a web page might have a feature "contains the word 'cat'".

- **Feature Column**: A set of related features, such as the set of all possible countries in which users might live. An example may have one or more features present in a feature column. "Feature column" is Google-specific terminology. A feature column is referred to as a "namespace" in the VW system (at Yahoo/Microsoft), or a field.
    - Example: An instance (with its features) and a label.

- **Model**: A statistical representation of a prediction task. You train a model on examples then use the model to make predictions.

- **Metric**: A number that you care about. May or may not be directly optimized.

- **Objective**: A metric that your algorithm is trying to optimize.

- **Pipeline**: The infrastructure surrounding a machine learning algorithm. Includes gathering the data from the front end, putting it into training data files, training one or more models, and exporting the models to production.

Abstract

Absenteeism is a habitual pattern of absence from a duty or obligation without good reason. Generally, absenteeism is unplanned absences. Absenteeism has been viewed as an indicator of poor individual performance, as well as a breach of an implicit contract between employee and employer. It is seen as a management problem, and framed in economic or quasi-economic terms. More recent scholarship seeks to understand absenteeism as an indicator of psychological, medical, or social adjustment to work.

While employers expect workers to miss a certain number of workdays each year, excessive absences can equate to decreased productivity and can have a major effect on company finances, morale and other factors.

Organizational Problems

- Cheerful, positive and encouraging
- Administrative costs of managing absenteeism
- Poor quality of goods/services resulting from overtime fatigue or understaffing
- Reduced productivity
- Excess manager time (dealing with discipline and finding suitable employee replacements)
- Safety issues (inadequately trained employees filling in for others, rushing to catch up after arriving as a replacement, etc)
- Wages paid to absent employees
- overtime pay for replacement employees and/or temporary workers
- Poor morale among employees who have to "fill in" or do extra work to cover absent coworkers

Goals

Absenteeism costs companies billions of dollars each year in lost productivity, wages, poor quality of goods/services and excess management time. In addition, the employees who do show up to work are often burdened with extra duties and responsibilities to fill in for absent employees, which can lead to feelings of frustration and a decline in morale.

- How much losses every month can we project in if same trend of absenteeism continues?
- What changes company should bring in to reduce the number of absenteeism?
    - If it is found that overall absenteeism is high amongst the employees the company should relook towards its policies likewise providing adequate paid/sick leaves.
    - If absenteeism is found in certain small % of employees then understand the root cause by discussing with target employees.
- To achieve a **Safe, Healthy & Competitive** working environment.
- Regular feedback of target employees

Problems to be Addressed

Data Acquisition

The first step to creating an effective plan is to set a baseline. The baseline is the foundation on which the other elements will be built on. This baseline could be termed as a set the problems which needs to be addressed. And based on this we will be **Fueling** our **Analysis**.

Non-Quantifiable Causes of Absenteeism:

- Illness
- Injuries
- Disengagement
- Bullying and harassment
- Depression
- Job hunting

Above are some of the problems an employee could be facing but the organization is unable to identify those, because of time bound or monetary limitations. Hence for a practical approach & to mark a viable difference we need to focus on parameters which are effortless to procure also can be computationally thrown to humans for analysis & also to algorithms to predict.

What could be those ?

To solve this problem organizations could implement a **Web** based system to collect data-points. These data-points could be easily fed to the system over a period of time so the organization needs to stay less invested towards collection of data. Also this could be completely automated by linking the Database with the company's Management Information System (MIS). To name a few data points which could turn useful in this case study are:

- Month of absence
- Service time
- Social smoker
- Distance from Residence to Work
- Transportation expense

***As data collection is out of scope for this project, above we have presented ways, by means of which business can implement this solution. For matter of this project we have resorted to "UCI Machine Learning Repository" for data acquisition.**

Scope of Work

This project entails analysis, visualizations, business logic, &interpretatinos.

ID	Reason for absence	Month of absence	Day of the week	Seasons	Transportation expense	Distance from Residence to Work	Service time	Age	Work load Average/day	Hit target	Disciplinary failure	Education	Son	Social drinker	Social smoker	Pet	Weight	Height	Body mass index	Absenteeism time in hours
11	26	7	3	1	289	36	13	33	2,39,554	97	0	1	2	1	0	1	90	172	30	4
36	0	7	3	1	118	13	18	50	2,39,554	97	1	1	1	1	0	0	98	178	31	0
3	23	7	4	1	179	51	18	38	2,39,554	97	0	1	0	1	0	0	89	170	31	2
7	7	7	5	1	279	5	14	39	2,39,554	97	0	1	2	1	1	0	68	168	24	4
11	23	7	5	1	289	36	13	33	2,39,554	97	0	1	2	1	0	1	90	172	30	2
3	23	7	6	1	179	51	18	38	2,39,554	97	0	1	0	1	0	0	89	170	31	2
10	22	7	6	1	361	52	3	28	2,39,554	97	0	1	1	1	0	4	80	172	27	8
20	23	7	6	1	260	50	11	36	2,39,554	97	0	1	4	1	0	0	65	168	23	4
14	19	7	2	1	155	12	14	34	2,39,554	97	0	1	2	1	0	0	95	196	25	40
1	22	7	2	1	235	11	14	37	2,39,554	97	0	3	1	0	0	1	88	172	29	8
20	1	7	2	1	260	50	11	36	2,39,554	97	0	1	4	1	0	0	65	168	23	8
20	1	7	3	1	260	50	11	36	2,39,554	97	0	1	4	1	0	0	65	168	23	8
20	11	7	4	1	260	50	11	36	2,39,554	97	0	1	4	1	0	0	65	168	23	8
3	11	7	4	1	179	51	18	38	2,39,554	97	0	1	0	1	0	0	89	170	31	1
3	23	7	4	1	179	51	18	38	2,39,554	97	0	1	0	1	0	0	89	170	31	4

The problem which we have taken in hand is **Multivariate** in nature.

What is "Multivariate"?

Multivariate data analysis is a set of statistical models that examine patterns in multidimensional data by considering, at once, several data variables. It is an expansion of bivariate data analysis, which considers only two variables in its models. As multivariate models consider more variables, they can examine more complex phenomena and find data patterns that more accurately represent the real world.

Keeping in mind our problem statement we further seggregate our variables as :

Dependent:

- Reason for absence
- Month of absence
- Day of the week
- Seasons
- Transportation expense
- Weight
- Distance from Residence to Work
- Service time
- Age
- Work load Average/day
- Height
- Hit target
- Disciplinary failure
- Education
- SonSocial drinker
- Social smoker
- Pet

Independent:

- Absenteeism time in hours

Exploratory Data Analysis

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 740 entries, 0 to 739
Data columns (total 21 columns):
```

We have 740 rows for 3 years of data.
So the entries in the data are only for absent days.

```
df.isnull().values.any()
```

```
False
```

No NA's to impute in the dataset.

```
df[df['Month of absence']==0]
```

	ID	Reason for absence	Month of absence	Day of the week	Seasons	Transportation expense
737	4	0	0	3	1	118
738	8	0	0	4	2	231
739	35	0	0	6	3	179

3 rows × 21 columns

3 rows in Month of Absence has "0" values.
This occurs to be corrupted data while looking at other column values in these rows. Hence we get rid of these rows.

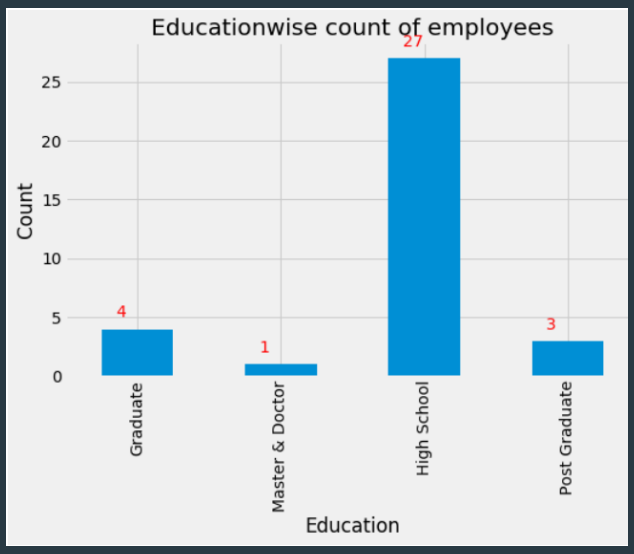
```
len(df[df['Absenteeism time in hours']==0])
```

```
41
```

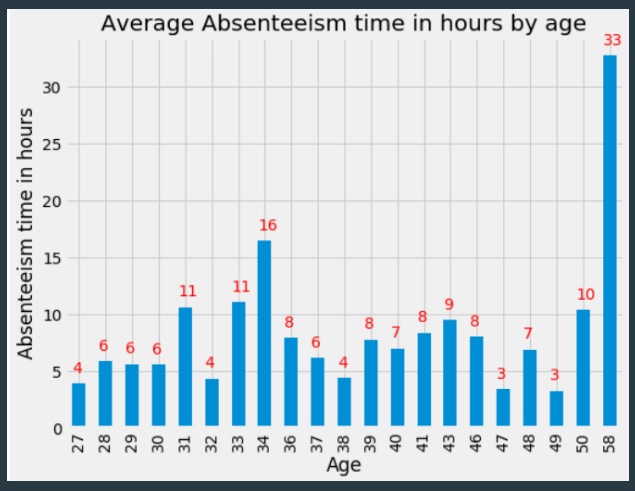
There are "41" "0's" in the target variable i.e Invalid values.

```
df.loc[(df['Absenteeism time in hours']==0),'Absenteeism time in hours']=8
```

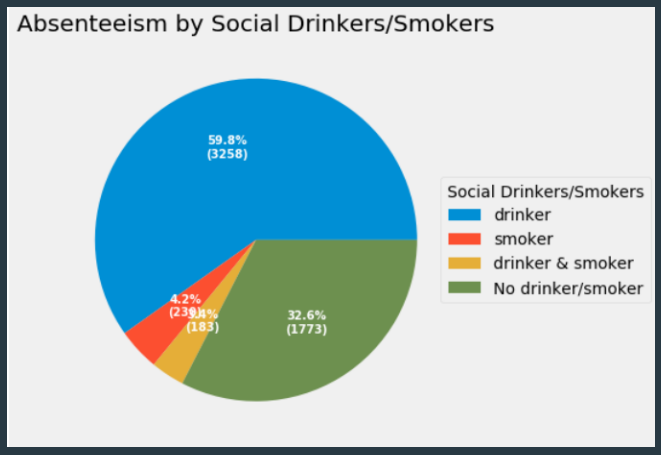
We replace this "0's" with 8 knowing there is 8 hours work culture followed across the globe.



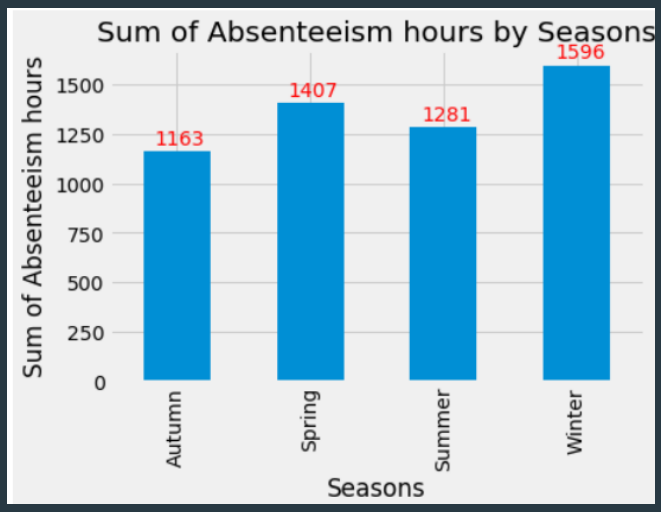
"High School" educated employees are higher than the rest.



Age group >50 has has the highest Absenteeism which is obvious considering the age. This could be interpreted as outlier.

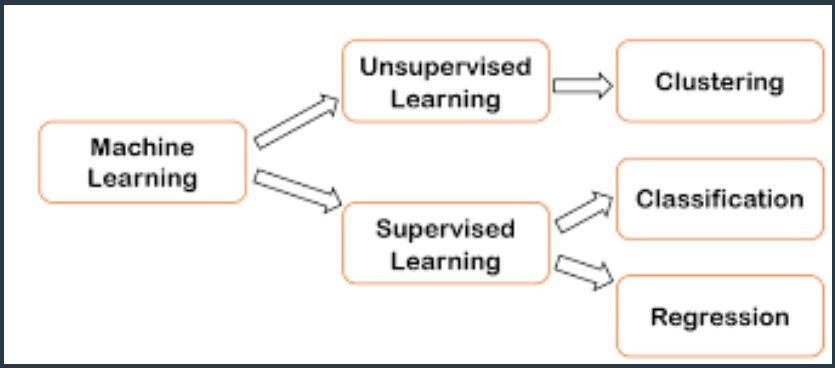


Drinkers account to highest % of Absenteeism.



Absenteeism in Winter proves to be highest as winter's are severe in Western Countries.

Modeling



To understand which types of algorithms can be applied on our data we take a look at the target variable & start from the root.

Machine Learning is broadly divided into 2 types :

- Supervised & Unsupervised
    - Supervised Learning :
        - Task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training example
    - Unsupervised Learning :
        - Type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses.

As we are provided with target variable "Absenteeism time in hours" we can infer our task under **Supervised Learning**.

Supervised Learning is further divided into 2 types :

- Regression & Classification

Classification Vs Regression

PARAMETER	CLASSIFICATION	REGRESSION
Prediction	The output variable is discrete in nature	The output variable is continuous in nature
Find	Decision boundary	Best Fit line
Output Data	Unordered	Ordered
Evaluation	calculate accuracy	Calculate sum of squared errors, R-squared
Example Algorithms	logistic regression, Decision Tree, Random Forest etc	Linear Regression, Polynomial Regression etc.

Regression :
Regression is a process of finding the correlations between dependent and independent variables. It helps in predicting the continuous variables such as prediction of Market Trends, prediction of House prices, etc.

Classification :
Classification is a process of finding a function which helps in dividing the dataset into classes based on different parameters. In Classification, a computer program is trained on the training dataset and based on that training, it categorizes the data into different classes.

Before identifying our problem as Regression or Classification lets have a look at our target variable.

Absenteeism time in hou
4
2
4
2
16
24
4
40
3
8
5
2
1

As we can see our target variable is continuous in nature we can apply Regression algorithms on this problem.

Also by applying a bit of feature engineering i.e by applying Bins to target variable this could be converted to a Classification problem.

We apply both the types on our data & evaluate between both using our evaluation metrics which are Accuracy, Precision & Recall.

We first divide our data in train for feeding the model & test for evaluating the model.

- Later on we apply both Regression & Classification Models.

Train Test Split

```
X = df_features
y = df['Absenteeism time in hours']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Regression

```
model = RandomForestRegressor(n_estimators=1000, random_state=0, oob_score=True, n_jobs=-1)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
```

We choose Random Forest Regression model.

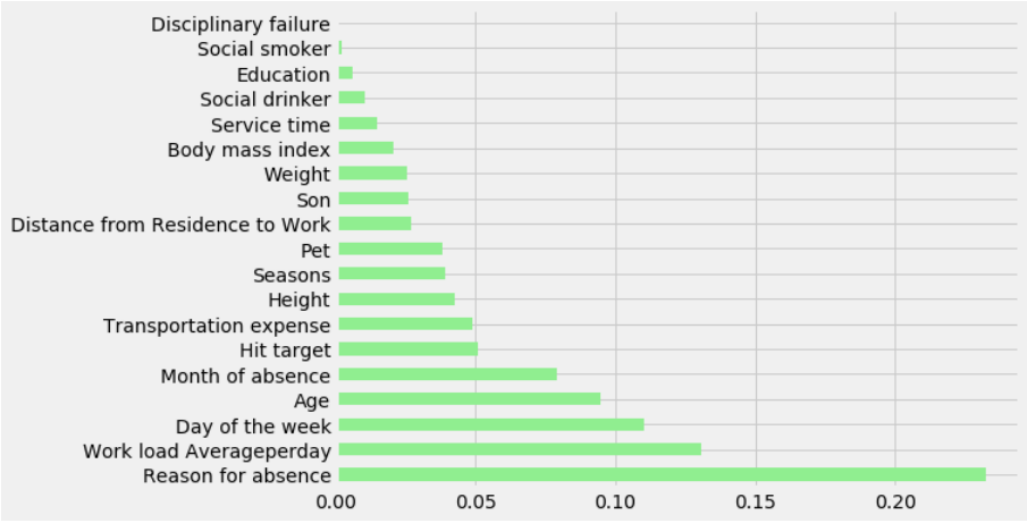
```
test_score = r2_score(y_test, y_pred)
spearman = spearmanr(y_test, y_pred)
pearson = pearsonr(y_test, y_pred)
rmse_test = MSE(y_test, y_pred) ** (1/2)

print(f'Out-of-bag score : {model.oob_score_:>5.3}')
print(f'Test data R-2 score: {test_score:>5.3}')
print(f'Test set of RMSE : {rmse_test:.2f}')
print(f'Test data Spearman correlation: {spearman[0]:.3}')
print(f'Test data Pearson correlation: {pearson[0]:.3}')
```

```
Out-of-bag score : -0.0335
Test data R-2 score: 0.0224
Test set of RMSE : 12.87
Test data Spearman correlation: 0.647
Test data Pearson correlation: 0.302
```

The negative out-of-bag score says the model is not interpreting results as expected.
R2 score is nearly zero the model is not predicting anything.

Visualizing the Random Forest Model



Classification

```
abs_bins = [0, 4, 8, 16, np.inf]
abs_names = ['<4', '8', '16', '16+']
df['abs_range'] = pd.cut(df['Absenteeism time in hours'], abs_bins, labels=abs_names)
```

We create bins of <4, 8, 16, 16+ to make the problem classification ready. Assumption that employee absence can be half day, full day, 2 days or more...

```
model = RandomForestClassifier(n_estimators=100, random_state=0, oob_score=True, n_jobs=-1)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
```

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

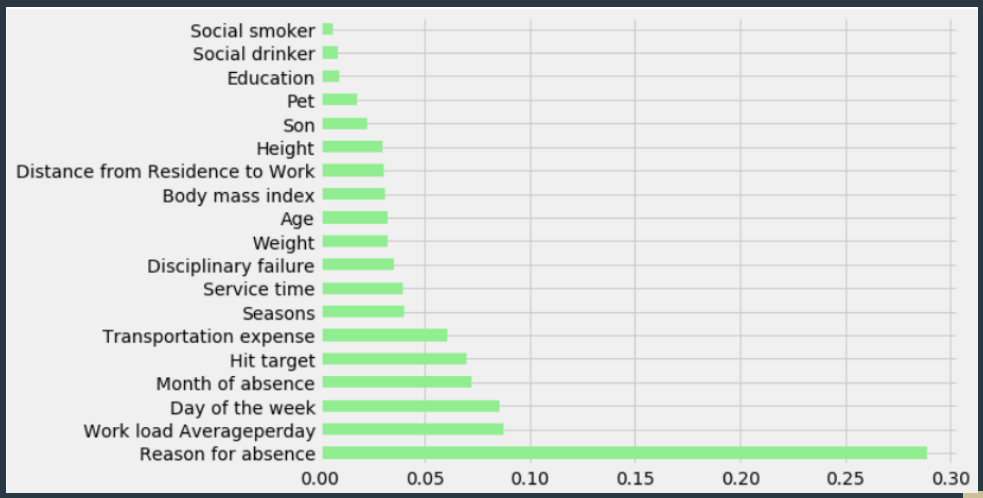
```
Accuracy: 0.7342342342342343
```

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
16	0.00	0.00	0.00	3
16+	0.33	0.18	0.24	11
8	0.66	0.64	0.65	78
<4	0.80	0.85	0.83	130
micro avg	0.73	0.73	0.73	222
macro avg	0.45	0.42	0.43	222
weighted avg	0.72	0.73	0.72	222

Above accuracy report is better compared to earlier regression model. The micro F1 score 0.73 is good compare to negative out-of-bag score. So we continue with Classification.

Visualizing the Random Forest Model



```
bins = [25, 35, 45, 55, np.inf]
names = [25, 35, 45, 55]
df['age_range'] = pd.cut(df['Age'], bins, labels=names)
```

```
X = df[['Reason for absence', 'Month of absence', 'Day of the week', 'Seasons', 'Transportati
        'Distance from Residence to Work', 'Service time', 'age_range', 'Work load Averageper
        'Disciplinary failure', 'Education', 'Son', 'Social drinker', 'Social smoker', 'Pet',
        'Height', 'Body mass index' ]]
y = df['abs_range']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

We have seen earlier during EDA that Age is also discrete. Let us convert the Age to Age Range.

```
model = RandomForestClassifier(n_estimators=100, random_state=0, oob_score=True, n_jobs=-1)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
```

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7432432432432432
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

          16       0.00      0.00      0.00         3
         16+       0.33      0.18      0.24        11
           8       0.68      0.63      0.65        78
          <4       0.80      0.88      0.84       130

   micro avg       0.74      0.74      0.74       222
   macro avg       0.45      0.42      0.43       222
weighted avg       0.72      0.74      0.73       222
```

The accuracy is improving. This shows that there is no end to Feature Engineering. And it is evolving process and we have to do better feature engineering to get the better results.

Applying Random Search to derive range of vaues.

```
from sklearn.model_selection import RandomizedSearchCV
n_estimators = [int(x) for x in np.linspace(start=100, stop = 2000, num = 20)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
min_samples_split = [2, 5, 10]
min_sample_leaf = [1, 2, 4]
bootstrap = [True, False]

random_grid = {'n_estimators' : n_estimators,
               'max_features' : max_features,
               'max_depth' : max_depth,
               'min_samples_split' : min_samples_split,
               'min_samples_leaf': min_sample_leaf,
               'bootstrap' : bootstrap
              }

rf = RandomForestClassifier()
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, n_iter = 1
                               verbose=2, random_state=42, n_jobs = -1)
rf_random.fit(X_train, y_train)
```

```
rf_random.best_params_
```

```
{'n_estimators': 300,
 'min_samples_split': 10,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 10,
 'bootstrap': True}
```

Now using this parameters we can set a range in Grid Search to evaluate best combinations.

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'bootstrap' : [True],
    'max_depth' : [8, 10, 12, 14],
    'max_features' : ['sqrt'],
    'min_samples_leaf' : [1, 3, 4],
    'min_samples_split' : [7, 10, 12],
    'n_estimators' : [250, 275, 300, 325]

}

rf = RandomForestClassifier()
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid, cv=5, n_jobs = -1, error_
```

Setting GridSearchCV.

```
grid_search.best_params_
```

```
{'bootstrap': True,
 'max_depth': 12,
 'max_features': 'sqrt',
 'min_samples_leaf': 4,
 'min_samples_split': 7,
 'n_estimators': 275}
```

```
from sklearn.ensemble import RandomForestClassifier
model_grid = grid_search.best_estimator_
model_grid.fit(X_train, y_train)
predictions_grid = model_grid.predict(X_test)
```

Fitting best_estimators to the model.

```
rf_random.best_params_
```

```
{'n_estimators': 300,
 'min_samples_split': 10,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 10,
 'bootstrap': True}
```

Now using this parameters we can set a range in Grid Search to evaluate best combinations.

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'bootstrap' : [True],
    'max_depth' : [8, 10, 12, 14],
    'max_features' : ['sqrt'],
    'min_samples_leaf' : [1, 3, 4],
    'min_samples_split' : [7, 10, 12],
    'n_estimators' : [250, 275, 300, 325]

}

rf = RandomForestClassifier()
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid, cv=5, n_jobs = -1, error_
```

Setting GridSearchCV.

```
grid_search.best_params_
```

```
{'bootstrap': True,
 'max_depth': 12,
 'max_features': 'sqrt',
 'min_samples_leaf': 4,
 'min_samples_split': 7,
 'n_estimators': 275}
```

```
from sklearn.ensemble import RandomForestClassifier
model_grid = grid_search.best_estimator_
model_grid.fit(X_train, y_train)
predictions_grid = model_grid.predict(X_test)
```

Fitting best_estimators to the model.

```
from sklearn.ensemble import RandomForestClassifier
model_grid = grid_search.best_estimator_
model_grid.fit(X_train, y_train)
predictions_grid = model_grid.predict(X_test)
```

```
print("Accuracy:",metrics.accuracy_score(y_test, predictions_grid))
```

```
Accuracy: 0.7477477477477478
```

```
print(classification_report(y_test, predictions_grid))
```

```
              precision    recall  f1-score   support

          16       0.00      0.00      0.00         3
         16+       1.00      0.09      0.17        11
           8       0.68      0.69      0.68        78
          <4       0.79      0.85      0.82       130

   micro avg       0.75      0.75      0.75       222
   macro avg       0.62      0.41      0.42       222
weighted avg       0.75      0.75      0.73       222
```

Now using this parameters we can set a range in Grid Search to evaluate best combinations.

Regression & Classification is a fundamental machine learning problem with applications across various products. In this project, we have broken down the regression & classification workflow into several steps. For each step, we have presented a detailed approach based on the characteristics of your specific dataset. By comparing between both the tyoes, we suggest a model type that gets you closer to the best performance quickly. The other steps such as "RandomizedSearchCV," "GridSearchCV" are engineered around this choice. We hope that following the project, the accompanying code, and the flowchart will help you learn, understand, and get a swift first-cut solution for similar business problems.

Timetable

DESCRIPTION OF WORK	START DATE	END DATE
PHASE 1 Team Build & Project Selection	15 Mar'20	20 Mar'20
PHASE 2 Analysis & Modeling	21 Mar'20	27 Mar'20
PHASE 3 Final Project Report	28 Mar'20	30 Mar'20

Signature:
Name: Aniket Patil
Project Advisor:

Signature:
Name: Raj Ahuja
Project Sponsor:

Signature:
Name: Yashraj Khokrale
Project Manager:

Signature:
Name: Prathamesh Shelar
Project Advisor: