COSC 3337 : Data Science I

N. Rizk

College of Natural and Applied Sciences

Department of Computer Science

University of Houston

Example Data Exploration +Analysis

Predicting why employees are leaving the company, and learn to predict who will leave the company.

Steps

- Employee Analysis
- Data loading and understanding feature
- Exploratory data analysis and Data visualization
- Cluster analysis
- Building prediction model using Gradient Boosting Tree.
- Evaluating model performance
- Conclusion

Exploratory Analysis

- Summarize characteristics of data such as pattern, trends, outliers, and hypothesis testing using #import modules, load file

```
import pandas  # for dataframes
import matplotlib.pyplot as plt # for plotting graphs
import seaborn as sns # for plotting graphs
% matplotlib inline descriptive statistics and
visualization.
data=pandas.read_csv('HR_comma_sep.csv')
data.head()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Departments	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium
3	0.72	0.87	5	223	5	0	1	0	sales	low
4	0.37	0.52	2	159	3	0	1	0	sales	low

data.tail()

Attributes names and datatypes using info().

data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 14,999 entries, 0 to 14,998

Data columns (total 10 columns):

```
satisfaction_level      14999 non-null float64
last_evaluation         14999 non-null float64
number_project          14999 non-null int64
average_montly_hours    14999 non-null int64
time_spend_company      14999 non-null int64
Work_accident           14999 non-null int64
left                    14999 non-null int64
promotion_last_5years   14999 non-null int64
Departments             14999 non-null object
salary                  14999 non-null object
```

dtypes: float64(2), int64(6), object(2)

memory usage: 1.1+ MB

This dataset has 14,999 samples, and 10 attributes(6 integer, 2 float, and 2 objects).

No variable column has null/missing values.

```
col_names =
hr.columns.tolist()
print("Column names:")
print(col_names)
print("\nSample data:")
hr.head()
```

Column names:
['satisfaction_level', 'last_evaluation', 'number_project', 'average_montly_hours', 'time_spend_company', 'Work_accident', 'left', 'promotion_last_5years', 'sales', 'salary'

Describe the 10 attributes

- Satisfaction_level: It is employee satisfaction point, which ranges from 0-1.
- last_evaluation: It is evaluated performance by the employer, which also ranges from 0-1.
- number_projects: How many numbers of projects assigned to an employee?
- average_monthly_hours: How many average numbers of hours worked by an employee in a month?
- time_spent_company: time_spent_company means employee experience. The number of years spent by an employee in the company.
- work_accident: Whether an employee has had a work accident or not.
- promotion_last_5 years: Whether an employee has had a promotion in the last 5 years or not.
- Departments: Employee's working department/division.
- Salary: Salary level of the employee such as low, medium and high.
- left: Whether the employee has left the company or not.

- Rename column name from “sales” to “department”
  `hr=hr.rename(columns = {'sales':'department'})`
- Print the types
  `hr.dtypes`
- data is pretty clean, no missing values?
  `hr.isnull().any()`
- Number of records and features
  `hr.shape`
- `hr['department'].unique()`
- combine “technical”, “support” and “IT” these three together and call them “technical”
  import numpy as np
- `hr['department']=np.where(hr['department'] =='support', 'technical', hr['department'])`
- `hr['department']=np.where(hr['department'] =='IT', 'technical', hr['department'])`

```
Satisfaction_level          False
last_evaluation             False
number_project              False
average_montly_hours        False
time_spend_company          False
Work_accident               False
left                        False
promotion_last_5years       False
department                  False
salary                      False
dtype: bool
```

Data Insights

- two types of employee one who stayed and another who left the company. So, you can divide data into two groups and compare their characteristics. Here, you can find the average of both the groups using groupby() and mean() function.
- `left = data.groupby('left')`
- `left.mean()`
- `hr['left'].value_counts()`

0 11428
1 3571
Name: left, dtype: int64

left	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_last_5years
0	0.666810	0.715473	3.786664	199.060203	3.380032	0.175009	0.026251
1	0.440098	0.718113	3.855503	207.419210	3.876505	0.047326	0.005321

Low Satisfaction

More Hours

Low promotion

Summary Statistics

- `data.describe()`

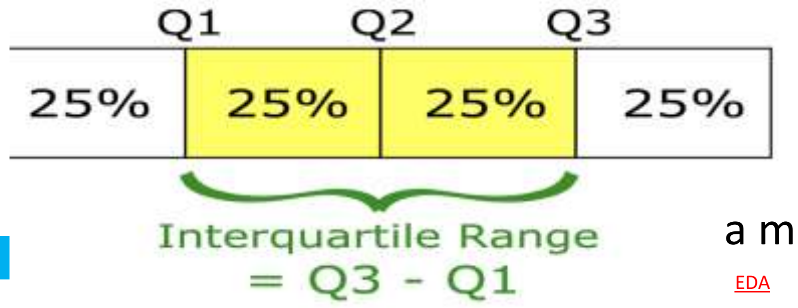
	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.238083	0.021268
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.425924	0.144281
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	0.000000
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000	0.000000
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000	0.000000
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000	1.000000

Central Tendency

The mean is simply the average

The mode is the value or category that occurs most often within the data

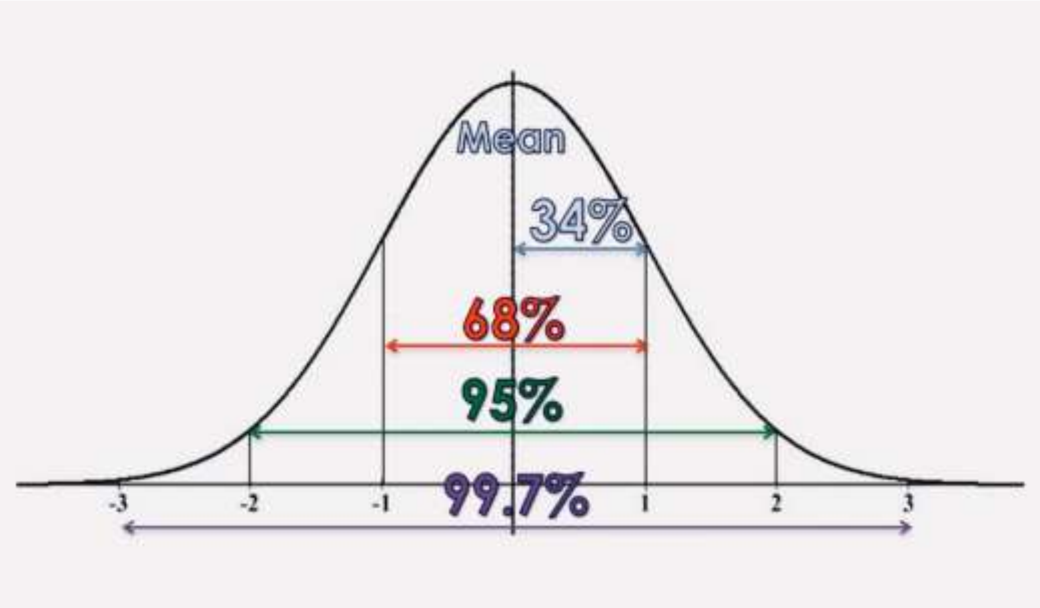
The median is the “middle” value or midpoint in your data



a measure of where the majority of the values lie.

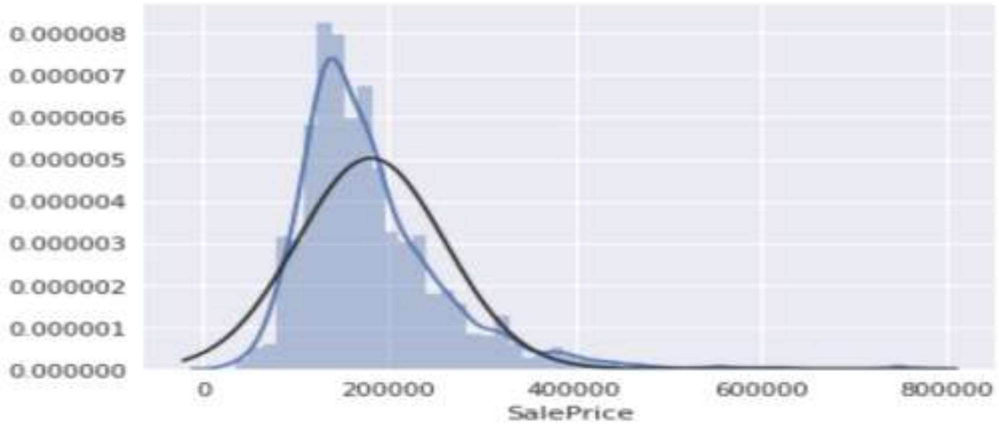
Summary Statistics…………….continue

- The Standard Deviation and the Variance measure the dispersion
  - low standard deviation=>data points close to the mean.
  - A high standard deviation =>data points are spread out
- Standard deviation is best used when data is unimodal (only one frequently occurring score, clustered at the top)

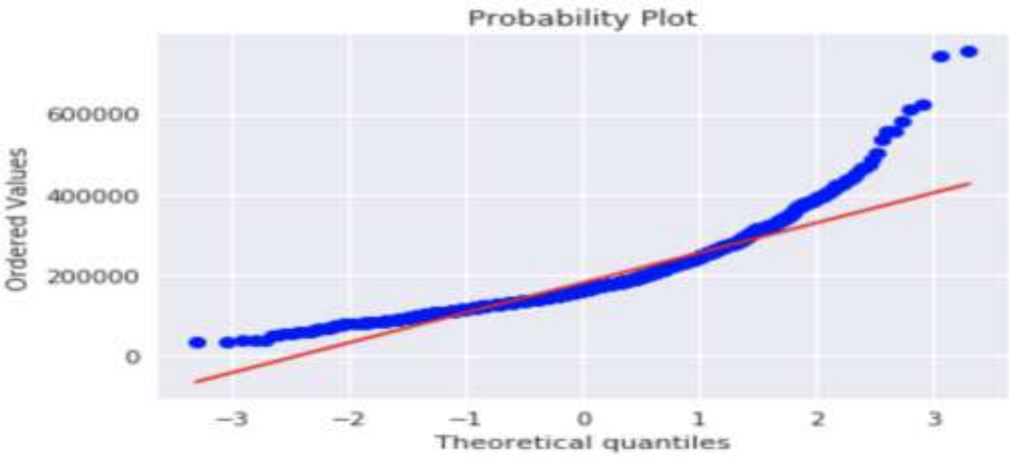


- Z-Score :how ma ean

- Skewness is a measurement of the symmetry of a distribution
- Kurtosis measures whether your dataset is heavy-tailed or light-tailed compared to a normal distribution



positively skewed dataset **.skew()**

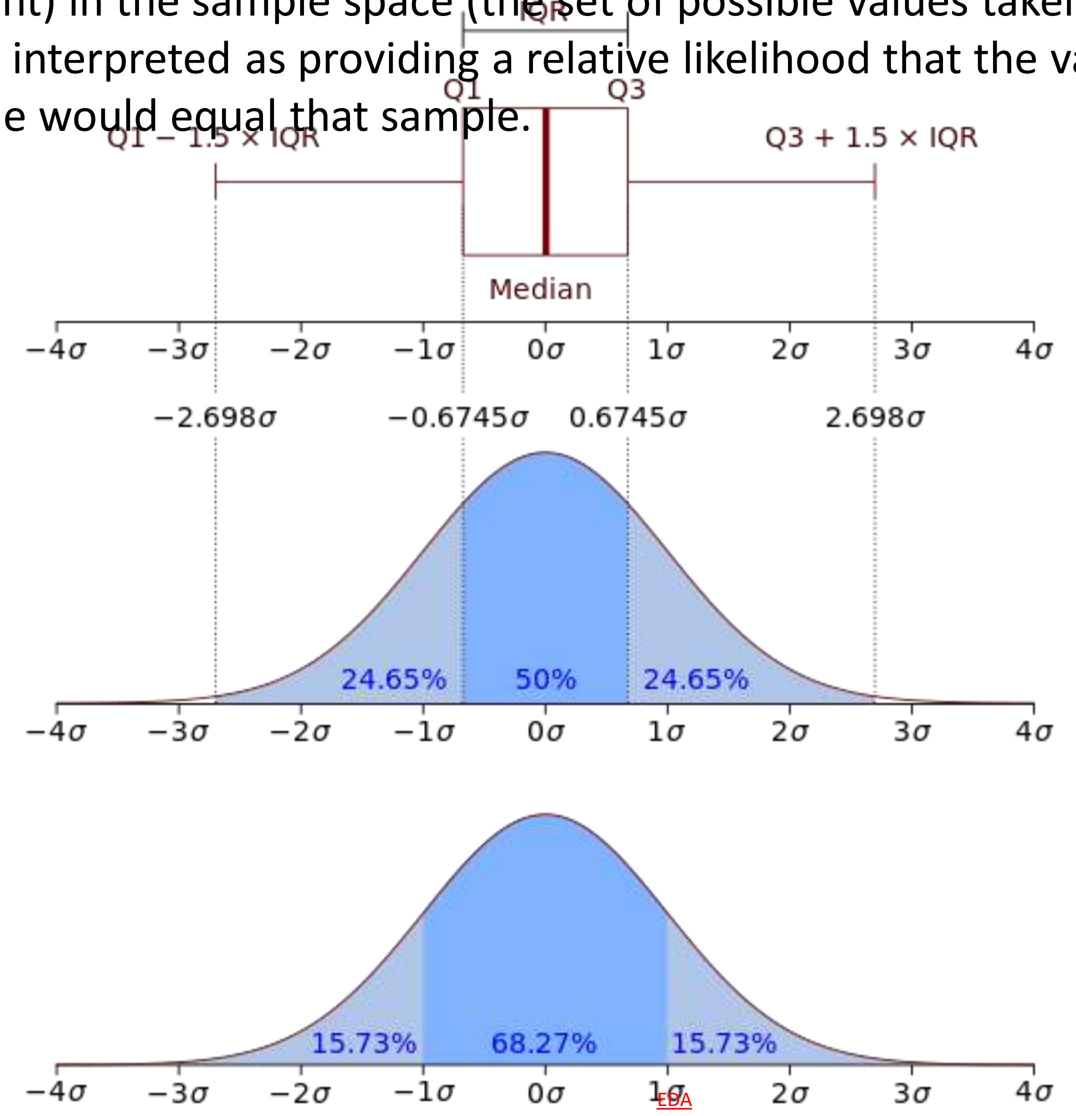


High kurtosis => more outliers

Probability density function curve is usually always 1, no matter what type of kurtosis

Probability density function

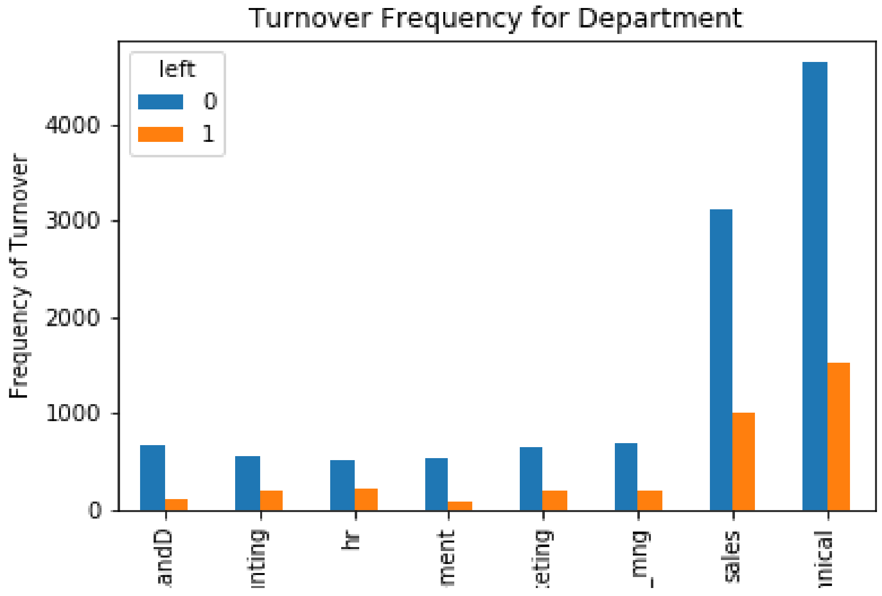
Density of a continuous random variable, is a function, whose value at any given sample (or point) in the sample space (the set of possible values taken by the random variable) can be interpreted as providing a relative likelihood that the value of the random variable would equal that sample.



Data Visualization

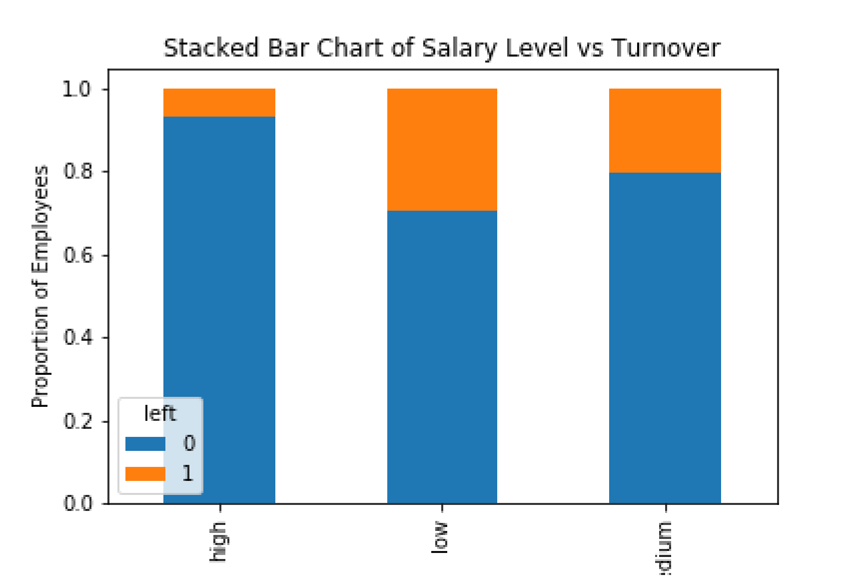
plot a bar graph using Matplotlib

```
%matplotlib inline
import matplotlib.pyplot as plt
pd.crosstab(hr.department,hr.left).plot(kind='bar')
plt.title('Turnover Frequency for Department')
plt.xlabel('Department')
plt.ylabel('Frequency of Turnover')
plt.savefig
```



the frequency of employee turnover depends a great deal on the department they work for. Thus, department can be a good predictor of the outcome variable.

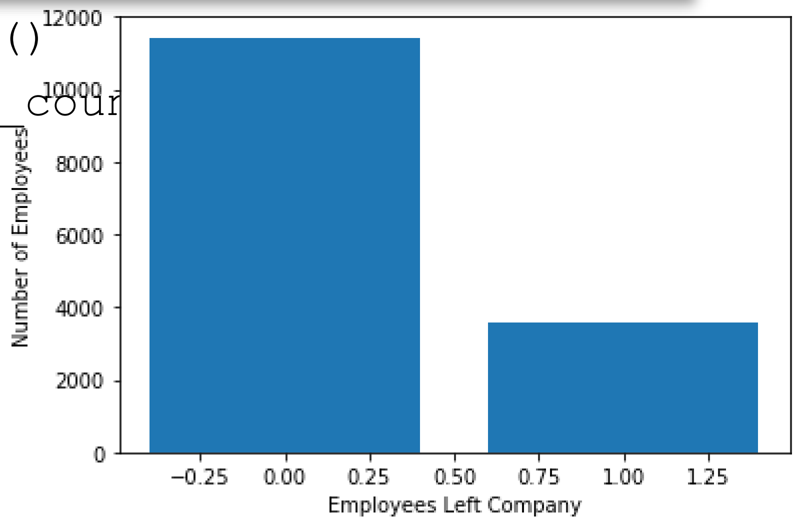
```
table=pd.crosstab(hr.salary, hr.left)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Salary Level vs Turnover')
plt.xlabel('Salary Level')
plt.ylabel('Proportion of Employees')
plt.savefig('salary_bar_chart')
```



The proportion of the employee turnover depends a great deal on their salary level; hence, salary level can be a good predictor in predicting the outcome.

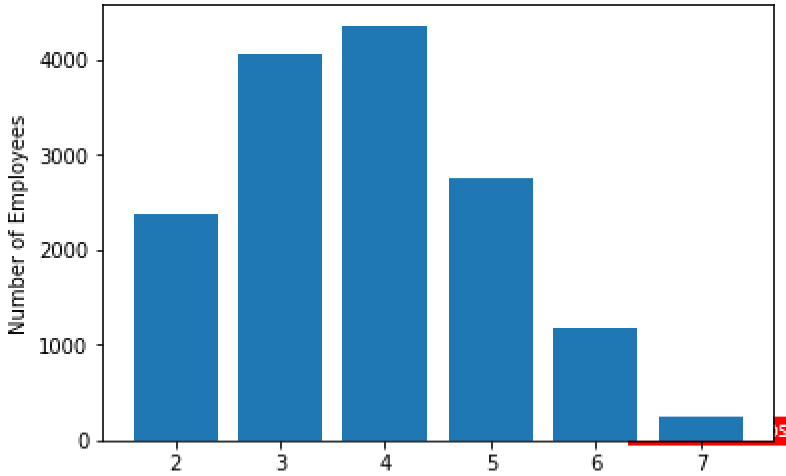
Plot using Matplotlib..continue

```
left_count=data.groupby('left').count()
plt.bar(left_count.index.values, left_cour
plt.xlabel('Employees Left Company')
plt.ylabel('Number of Employees')
plt.show()

data.left.value_counts()
```



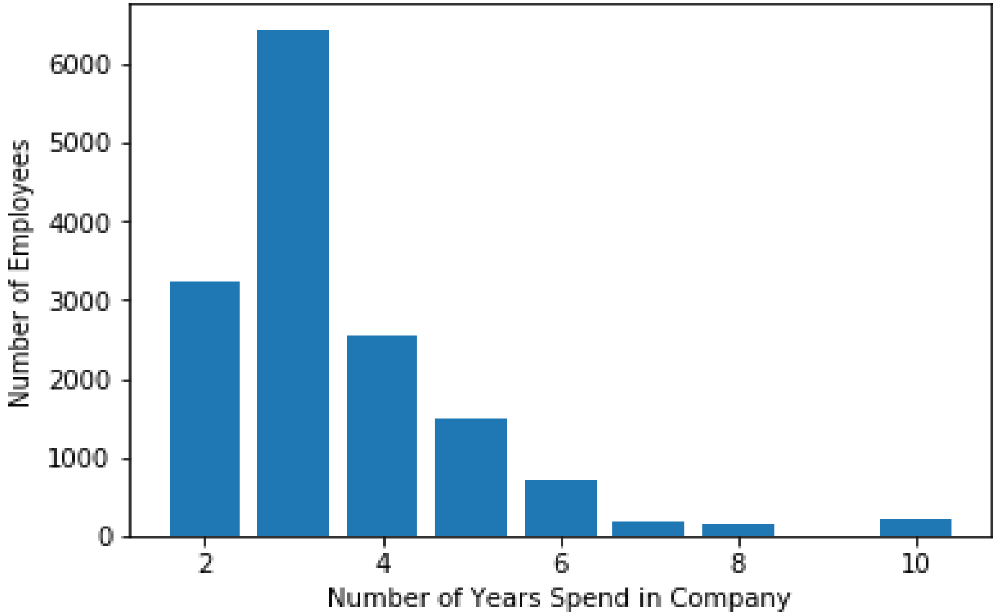
out of 15,000 approx 3,571 were left, and 11,428 stayed. The no of employee left is 23 % of the total employment.

```
num_projects=data.groupby('number_project').count()
plt.bar(num_projects.index.values, num_projects['satisfaction_level'])
plt.xlabel('Number of Projects')
plt.ylabel('Number of Employees')
plt.show()
```



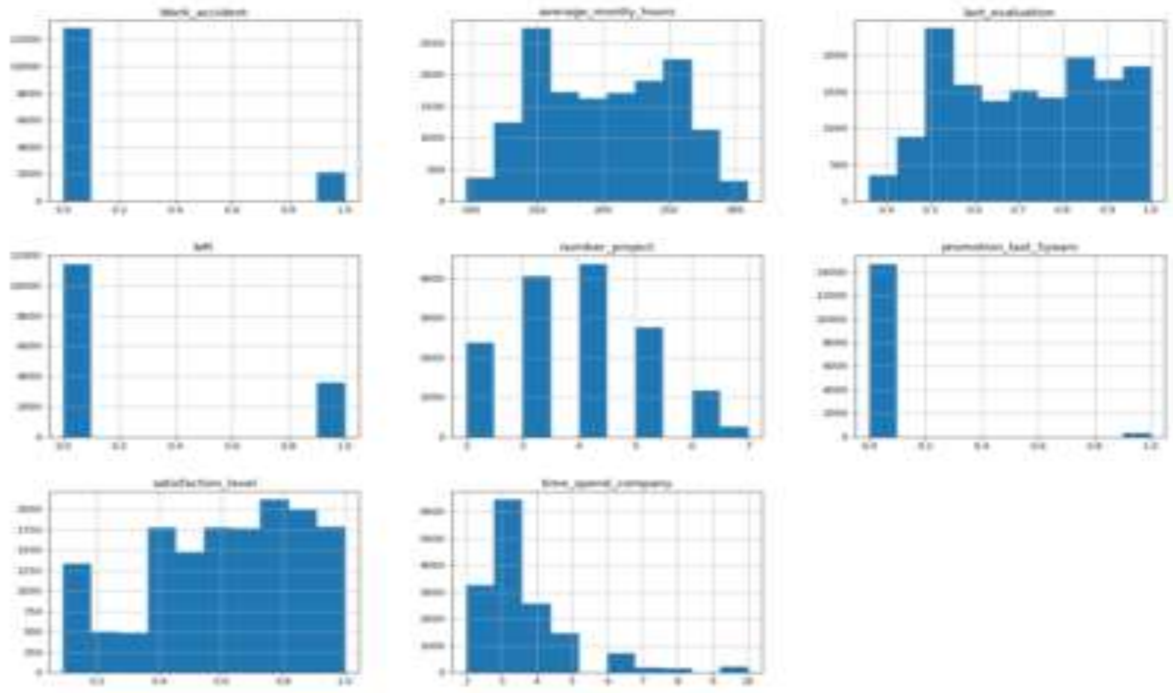
Plot using Matplotlib..continue

```
time_spent=data.groupby('time_spend_company').count()
plt.bar(time_spent.index.values, time_spent['satisfaction_level'])
plt.xlabel('Number of Years Spend in Company')
plt.ylabel('Number of Employees')
plt.show()
```



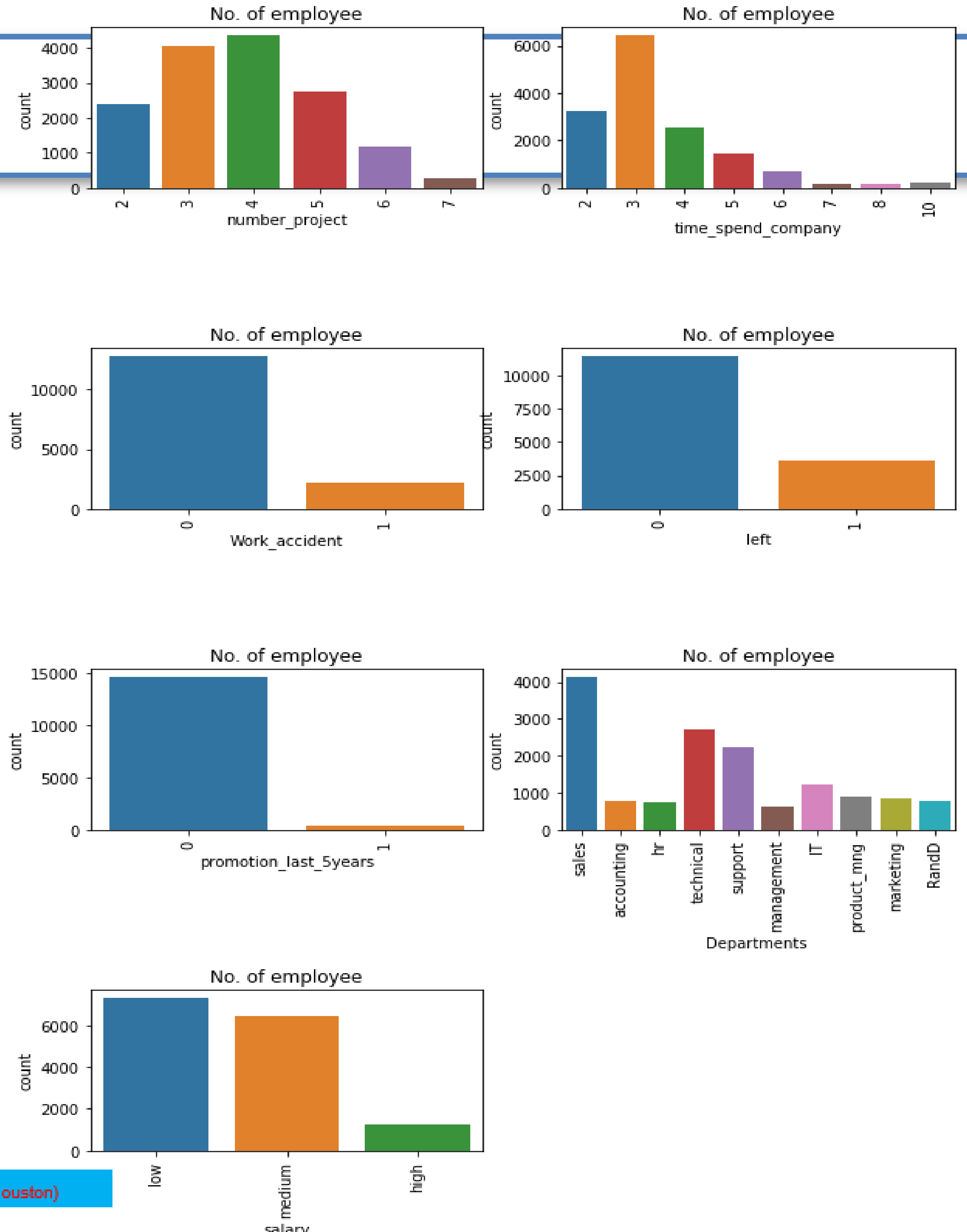
Plot using Matplotlib..continue

```
num_bins = 10
hr.hist(bins=num_bins, figsize=(20,15))
plt.savefig("hr_histogram_plots")
plt.sh
```



Subplots using Seaborn:: plot all graphs at once

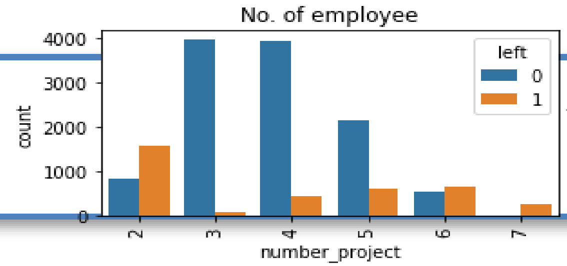
```
features=['number_project','time_spend_company','Work_accident','left',
'promotion_last_5years','Departments','salary']

fig=plt.subplots(figsize=(10,15))

for i, j in enumerate(features):

    plt.subplot(4, 2, i+1)

    plt.subplots_adjust(hspace = 1.0)

    sns.countplot(x=j,data = data)

    plt.xticks(rotation=90)

    plt.title("No. of employee")
```

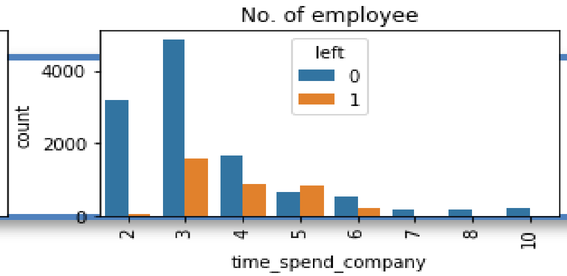


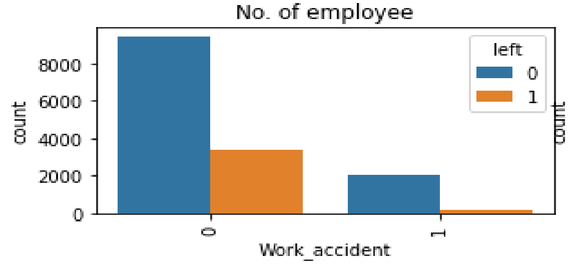
- Most of the employee is doing the project from 3-5.
- There is a huge drop between 3 years and 4 years experienced employee.
- The no of employee left is 23 % of the total employment.
- A decidedly less number of employee get the promotion in the last 5 year.
- The sales department is having maximum no.of employee followed by technical and support
- Most of the employees are getting salary either medium or low.

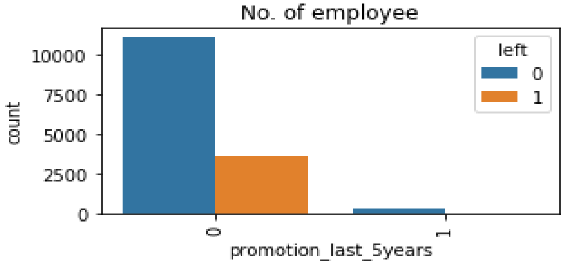
Seaborn…continue

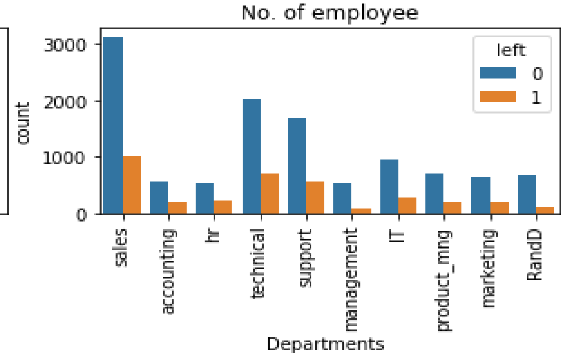
```
fig=plt.subplots(figsize=(10,15))
for i, j in enumerate(features):
    plt.subplot(4, 2, i+1)
    plt.subplots_adjust(hspace = 1.0)
    sns.countplot(x=j,data = data, hue='left')
    plt.xticks(rotation=90)
    plt.title("No. of employee")
```

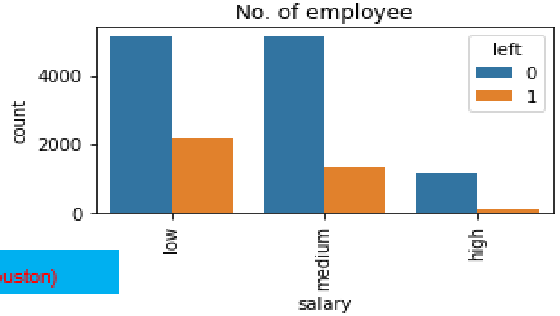












- Those employees who have the number of projects more than 5 were left the company.

- The employee who had done 6 and 7 projects, left the company it seems to like that they were overloaded with work.

- The employee with five-year experience is leaving more because of no promotions in last 5 years and more than 6 years experience are not leaving because of affection with the company.

- Those who promotion in last 5 years they didn't leave, i.e., all those left they didn't get the promotion in the previous 5 years.

Data Analysis and Visualization Summary:

- Promotions: Employees are far more likely to quit their job if they haven't received a promotion in the last 5 years.

- Time with Company: Here, The three-year mark looks like a time to be a crucial point in an employee's career. Most of them quit their job around the three-year mark. Another important point is 6-years point, where the employee is very unlikely to leave.

- Number Of Projects: Employee engagement is another critical factor to influence the employee to leave the company. Employees with 3-5 projects are less likely to leave the company. The employee with less and more number of projects are likely to leave.

- Salary: Most of the employees that quit among the mid or low salary groups.

Cluster Analysis: based on satisfaction and performance

```
#import module
from sklearn.cluster import KMeans
# Filter data
left_emp =  data[['satisfaction_level', 'last_evaluation']][data.left == 1]
# Create groups using K-means clustering.
kmeans = KMeans(n_clusters = 3, random_state = 0).fit(left_emp)
# Add new column "label" annd assign cluster labels.
left_emp['label'] = kmeans.labels_
# Draw scatter plot
plt.scatter(left_emp['satisfaction_level'], left_emp['last_evaluation'],
c=left_emp['label'],cmap='Accent')
plt.xlabel('Satisfaction Level')
plt.ylabel('Last Evaluation')
plt.title('3 Clusters of employees who left')
plt.show()
```

Employee who left the company can be grouped into 3 type of employees

- High Satisfaction and High Evaluation(Shaded by green color in the graph). Winners.
- Low Satisfaction and High Evaluation(Shaded by blue color), Frustrated.
- Moderate Satisfaction and moderate Evaluation (Shaded by grey color in the graph), 'Bad match'.

Creating Dummy Variables for Categorical Variables

```
cat_vars=['department','salary']
for var in cat_vars:
    cat_list='var'+'_'+var
    cat_list = pd.get_dummies(hr[var],
prefix=var)
    hr1=hr.join(cat_list)
    hr=hr1
```

The actual categorical variable needs to be removed once the dummy variables have been created.

```
hr.drop(hr.columns[[8, 9]], axis=1, inplace=True)
hr.columns.values
```

array(['satisfaction_level', 'last_evaluation',
'number_project',
'average_montly_hours',
'time_spend_company', 'Work_accident',
'left', 'promotion_last_5years',
'department_RandD',
'department_accounting',
'department_hr', 'department_manageme
'department_marketing',
'department_product_mng',
'department_sales',
'department_technical', 'salary_high',
'salary_low', 'salary_medium'],
dtype=object)

Left is the outcome, all the others predictors

```
hr_vars=hr.columns.values.tolist
()
y=['left']
X=[i for i in hr_vars if i not
in y]
```

Label encoding of categorical using sklearn

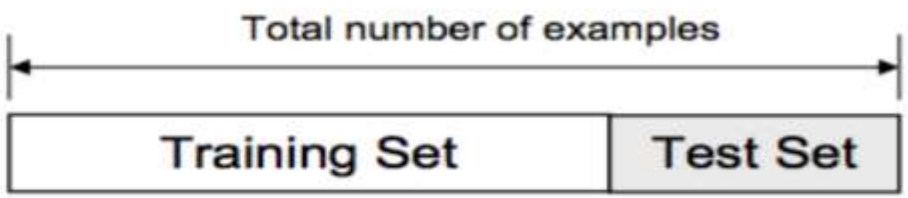
Convert salary and department to numerical

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Departments	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium
3	0.72	0.87	5	223	5	0	1	0	sales	low
4	0.37	0.52	2	159	3	0	1	0	sales	low

```
# Import LabelEncoder
from sklearn import preprocessing
#creating labelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers.
data['salary']=le.fit_transform(data['salary'])
data['Departments ']=le.fit_transform(data['Departments '])
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Departments	salary
0	0.38	0.53	2	157	3	0	1	0	7	1
1	0.80	0.86	5	262	6	0	1	0	7	2
2	0.11	0.88	7	272	4	0	1	0	7	2
3	0.72	0.87	5	223	5	0	1	0	7	1
4	0.37	0.52	2	159	3	0	1	0	7	1

Split Train and Test Set




```
#Spliting data into Feature and
X=hr[['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident',
       'promotion_last_5years', 'Departments ', 'salary']]
y=hr['left']

# Import train_test_split function
from sklearn.model_selection import train_test_split


# Split dataset into training set and test set
# 70% training and 30% test


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

```
print( X_train.shape, y_train.shape)
print (X_test.shape, y_test.shape)
```

(10499, 9) (10499,)
(4500, 9) (4500,)

Fit the model on the training data

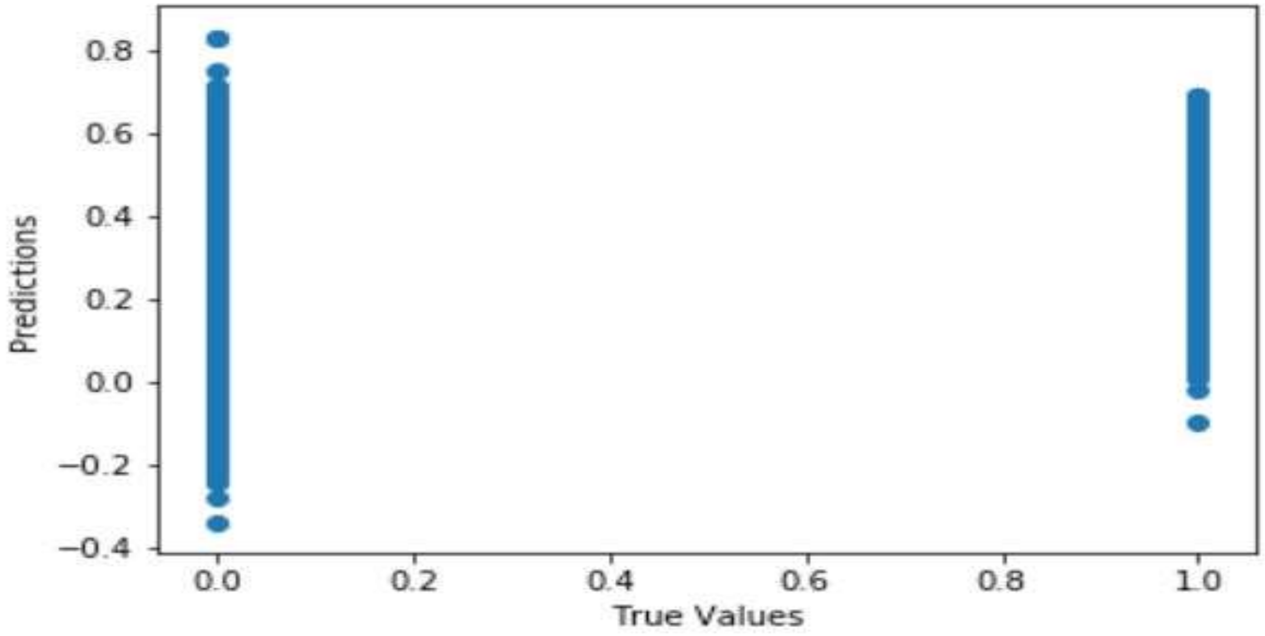
```
# fit a model
from sklearn import datasets, linear_model
lm = linear_model.LinearRegression()
model = lm.fit(X_train, y_train)
predictions = lm.predict(X_test)

Print the first 5 predictions
predictions[0:5]
```

array([0.03642494, 0.01215191, 0.22603346, 0.39069954, -0.12450459])

fitting the model on the training data and trying to predict the test data.

plot the model:

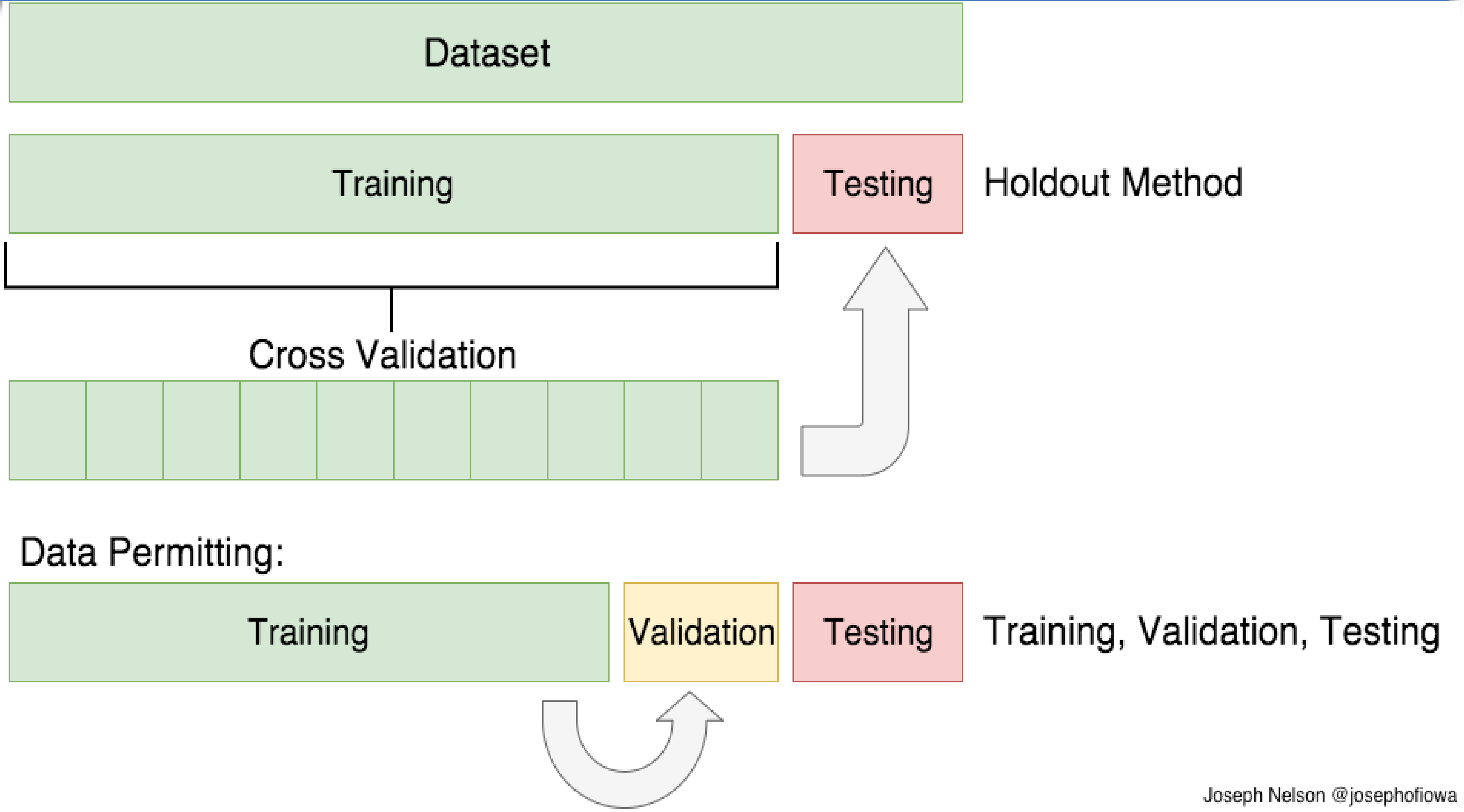
```python
## The line / model
from matplotlib import pyplot as plt
plt.scatter(y_test, predictions)
plt.xlabel("True Values")
plt.ylabel("Predictions")
plt.show()
```



print("Score:", model.score(X_test, y_test))

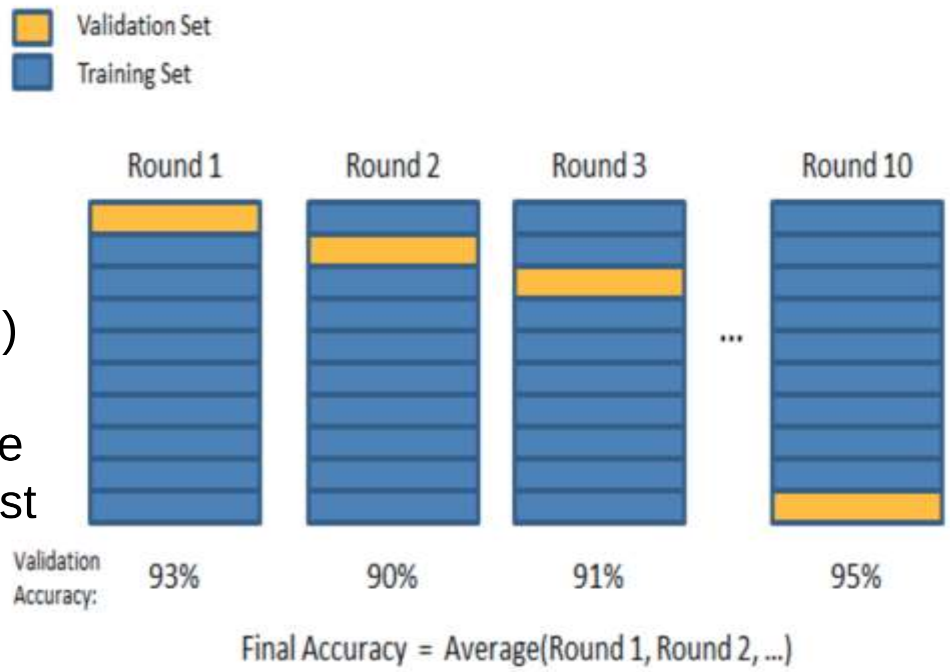
```
Score: 0.18388775485387 51
```

Cross Validation



Split the data into k subsets, and train on k-1:
one of those subset

K-Folds Cross Validation



Split the data into k different subsets (or folds). Use k-1 subsets to train our data and leave the last subset (or the last fold) as test data. Then average the model against each of the folds and then finalize the model. After that test it against the test set.

```
from sklearn.model_selection import KFold # import KFold
X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]]) # create an array
y = np.array([1, 2, 3, 4]) # Create another array
kf = KFold(n_splits=2) # Define the split - into 2 folds
kf.get_n_splits(X) # returns the number of splitting iterations in the cross-validator
print(kf)
KFold(n_splits=2, random_state=None, shuffle=False)
```

```
for train_index, test_index in kf.split(X):
 print("TRAIN:", train_index, "TEST:", test_index)
 X_train, X_test = X[train_index], X[test_index]
 y_train, y_test = y[train_index], y[test_index]
('TRAIN:', array([2, 3]), 'TEST:', array([0, 1]))
('TRAIN:', array([0, 1]), 'TEST:', array([2, 3])
```

Validation

```
# Necessary imports:
from sklearn.cross_validation import
cross_val_score, cross_val_predict
from sklearn import metrics
```

```
# Perform 6-fold cross validation
scores = cross_val_score(model, df, y, cv=6)
print “Cross-validated scores:”, scores
Cross-validated scores: [ 0.4554861
0.46138572  0.40094084  0.55220736
0.43942775  0.56923406]
```

```
# Make cross validated predictions
predictions = cross_val_predict(model, df, y,
cv=6)
plt.scatter(y, predictions)
```

```
accuracy = metrics.r2_score(y, predictions)
print “Cross-Predicted Accuracy:”, accuracy
Cross-Predicted Accuracy: 0.490806583864
```