Which employees will leave next?

Yun Choi

Table of Contents

Problem Statement and Hypothesis

- Data: 9 explanatory variables and outcome variable whether the employee has left for 15,000 current and former employees. (Source: Kaggle)

- Problem Statement: Determine which employees will leave using employee data from Kaggle which include salary level, satisfaction level, last evaluation, average monthly hours, etc. (Classification problem)

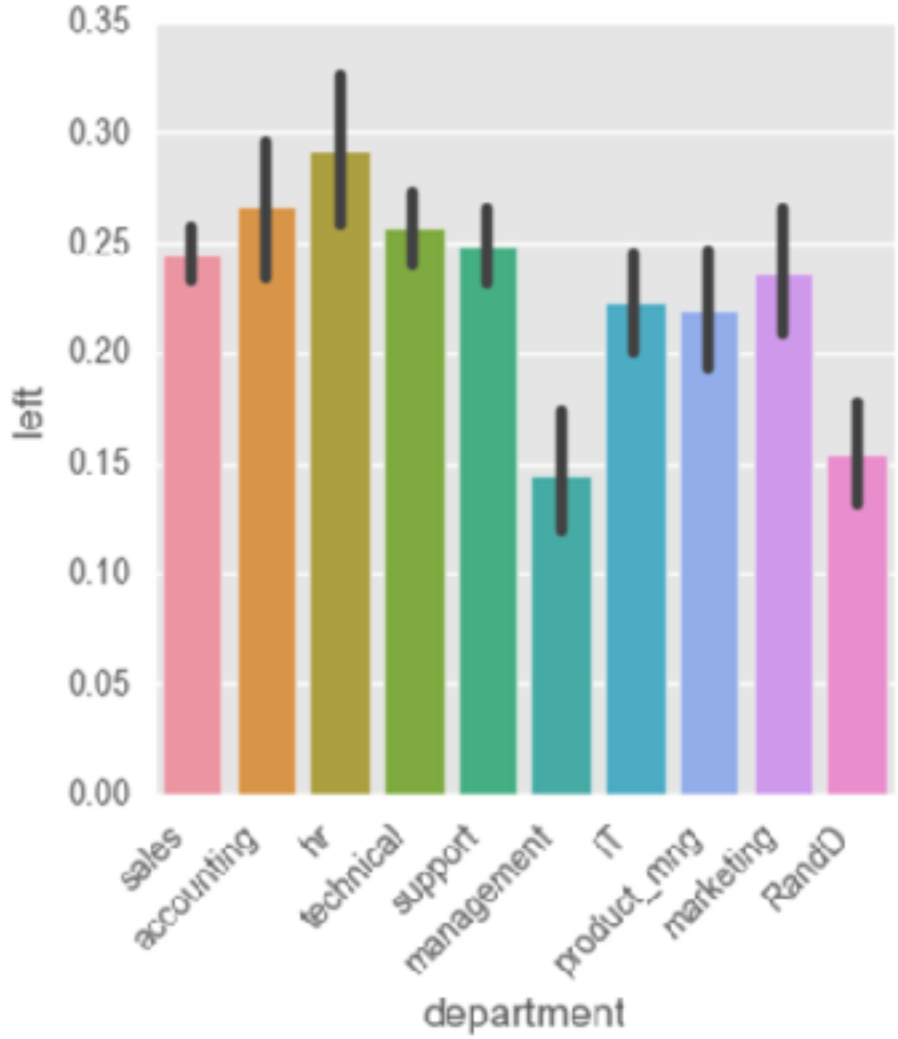
  - Different types of employees leaving (clustering problem)

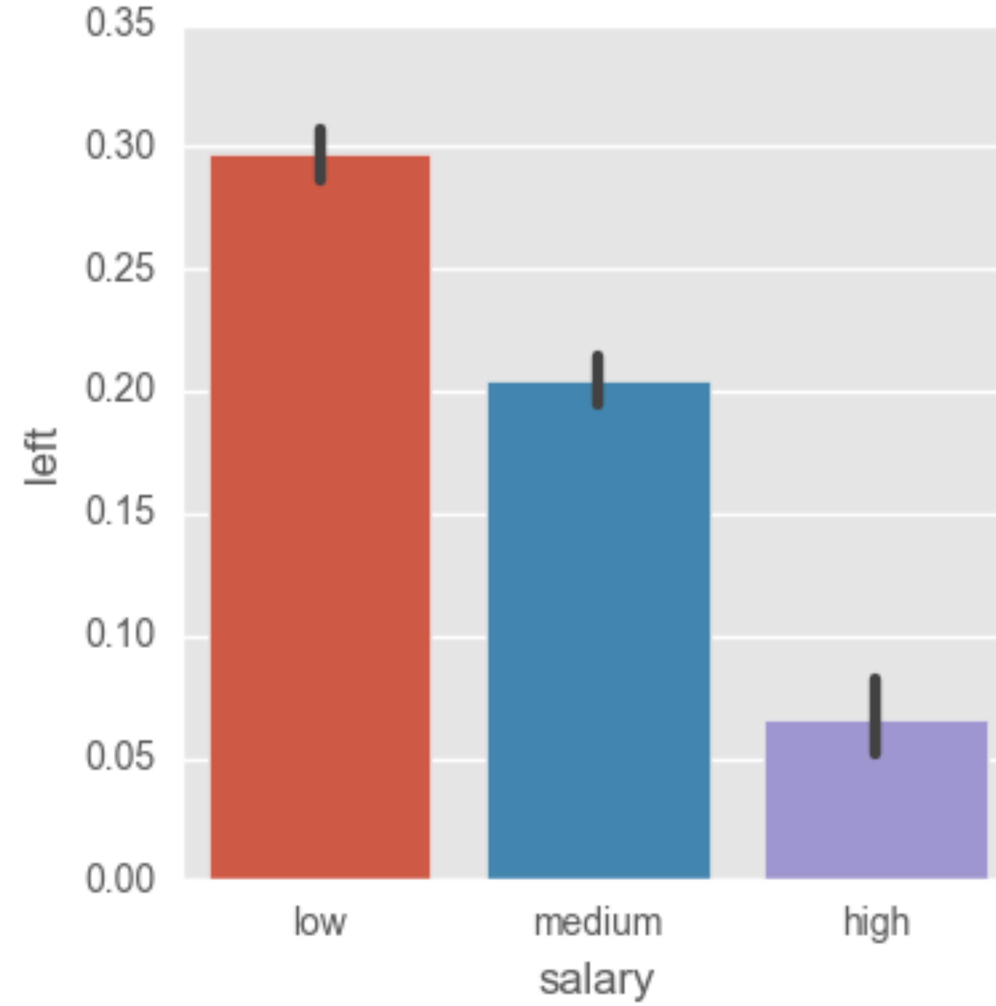
- Hypothesis: Employee data will help predict which employees will leave the company.

Data Dictionary

Variable	Description	Type of Variable	Range
satisfaction_level	Satisfaction level of employee based on survey	Continuous	[0.09, 1]
last_evaluation	Score based on employee's last evaluation	Continuous	[0.36, 1]
number_project	Number of projects	Continuous	[2, 7]
average_monthly_hours	Average monthly hours	Continuous	[96, 310]
time_spend_company	Years at company	Continuous	[2, 10]
Work_accident	Whether employee had a work accident	Categorical	{0, 1}
left	Whether employee had left (Outcome Variable)	Categorical	{0, 1}
promotion_last_5years	Whether employee had a promotion in the last 5 years	Categorical	{0, 1}
department	Department employee worked in	Categorical	10 departments
salary	Level of employee's salary	Categorical	{low, medium, high}

EDA

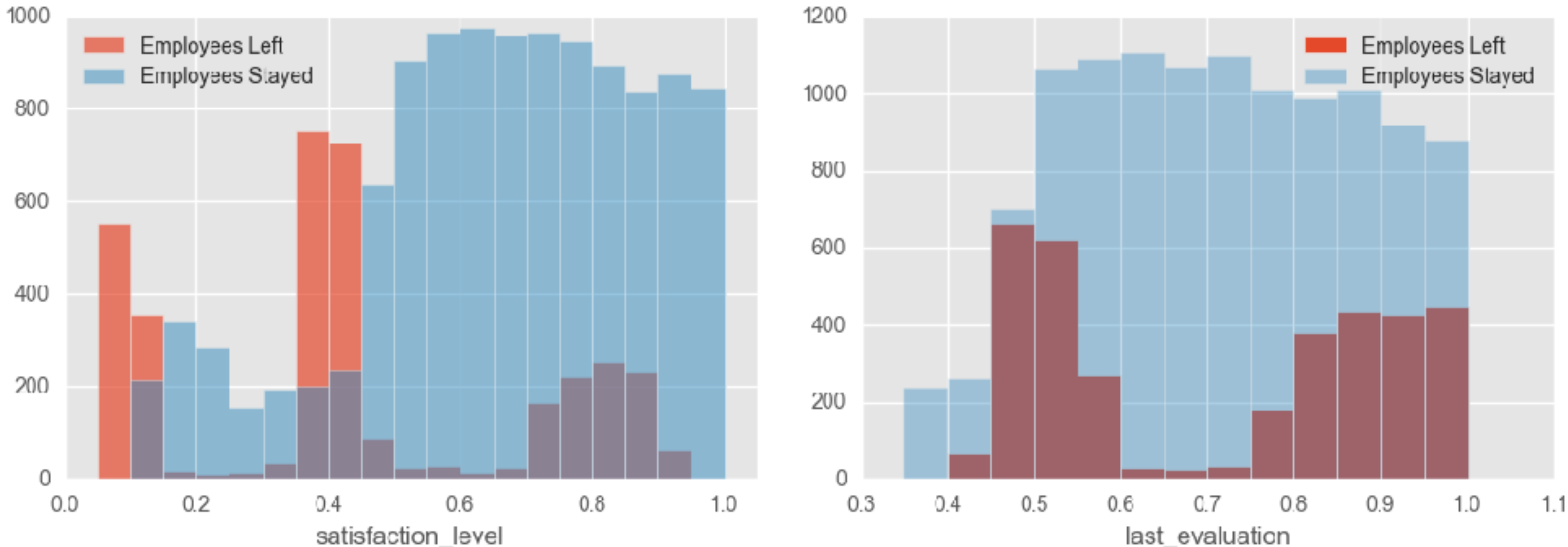




- Variables department and salary need to be converted to dummy variables using one-hot encoding.

- Created two dummy variables for salary (high, medium) and two dummy variables (management, R&D). The remaining 8 departments have similar employee attrition rate.

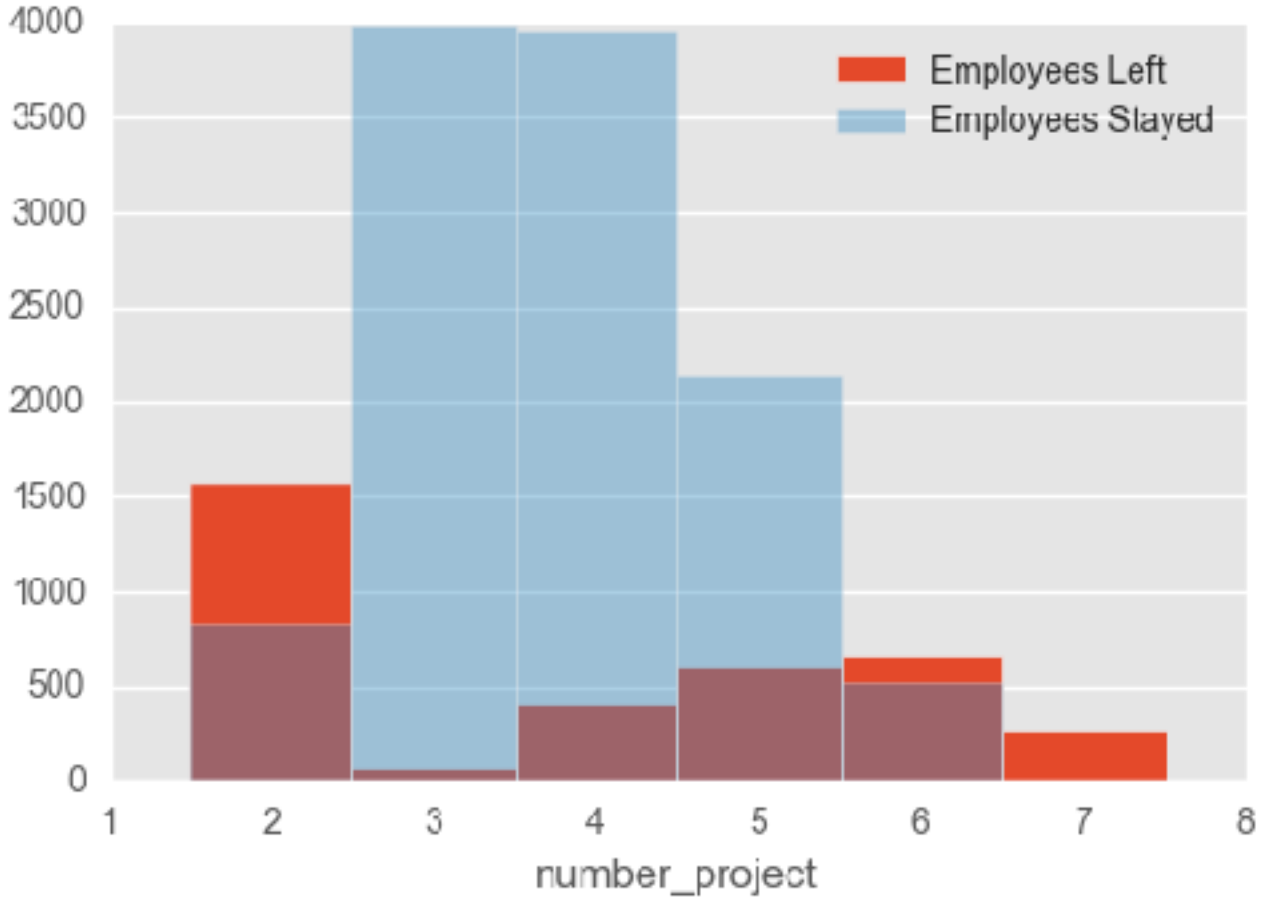
EDA

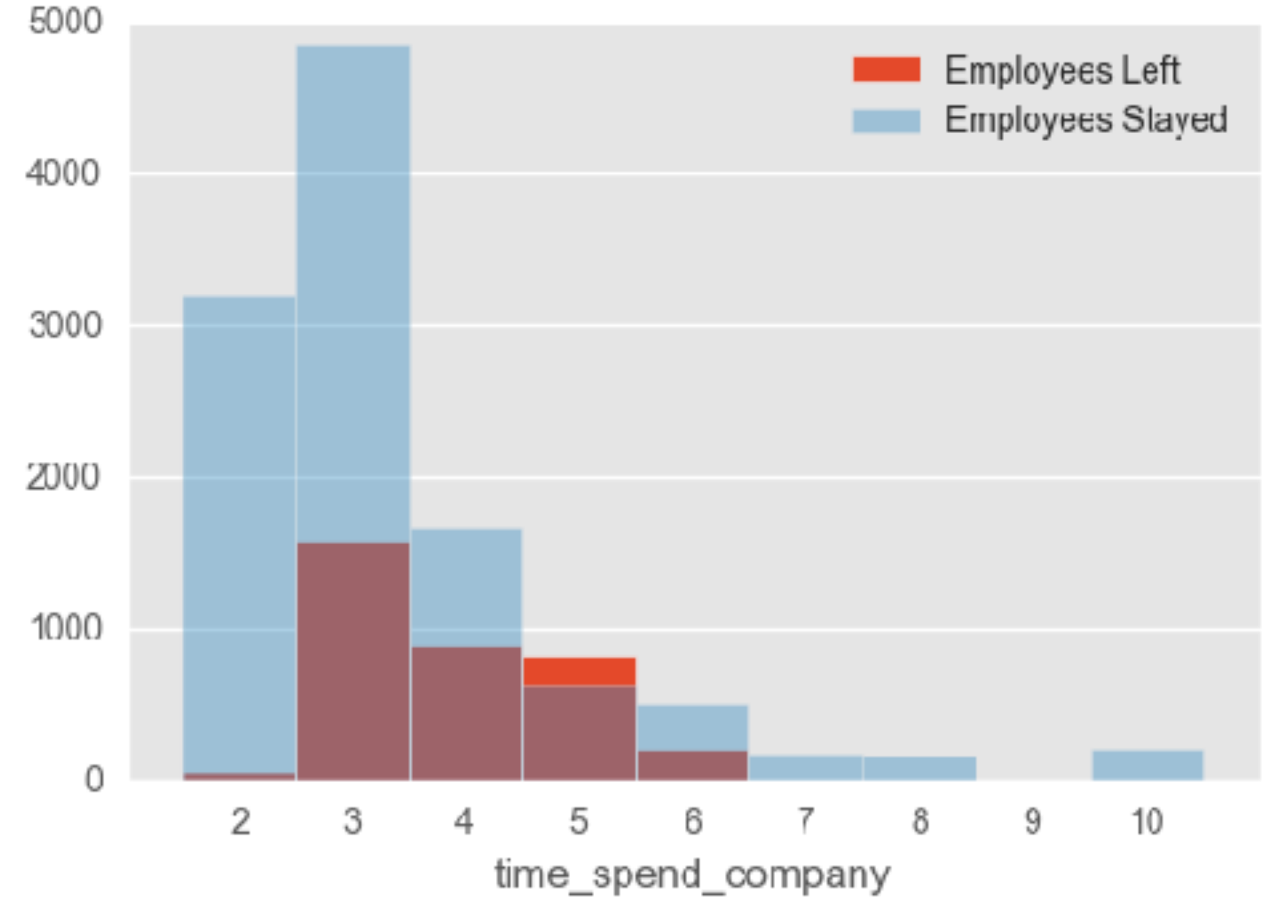


- Satisfaction level: 3 clusters of employees leaving (satisfied, below average and disgruntled)

- Last evaluation: two large groups of former employees, high performing group and poorly performing group, explains the nearly zero correlation between last_evaluation and the outcome variable.

EDA





- Number of projects: too few projects = bad, too many projects = bad. Sweet spot is in the 3-4 range.

- Years at company: nobody left after working 7 years or longer in the company. Employees in the year 5 group have the highest flight risk.

- Other variables: very low promotion rate (2% in last five years), average monthly hours graph look similar to number of projects.

Methods and Models

- Data
  - 70/30 split for training set and test set (train_test_split)
  - Feature scaling (StandardScaler) required for some learners
  - Cross Validation using ShuffleSplit(n_splits=20, test_size=0.3)

- Learning Algorithms
  - K-Nearest Neighbors
  - Random Forest
  - Logistic Regression
  - K-Means Clustering

- Metric: accuracy is what we want to optimize. Bonus, if we can explain how each feature contributes or calculate probabilities.

- Kitchen Sink Strategy: throw all variables in first, then subtract.

K-Nearest Neighbors

- Simple model to start out; number of features small enough (11 variables including 4 dummy); will lack interpretation

- Feature scaling used for kNN

- GridSearchCV to find best parameters
{'n_neighbors': 8, 'weights': ‘distance'}

```
from sklearn.model_selection import GridSearchCV
parameters = {'n_neighbors': range(1,11), 'weights': ['uniform', 'distance']}
clf = GridSearchCV(knn, parameters, cv=cv)
clf.fit(X_train_std, y_train)
clf.best_params_
```

- Result: 98% accuracy on test data. 76% is lower bound.

```
best_knn = clf.best_estimator_
```

```
best_knn.score(X_test_std, y_test)
```

```
0.97599999999999998
```

Random Forest

- KNN had great accuracy but RF will help us determine which features are important
- Same features as before (no scaling needed); GridSearchCV to find best parameters {'n_estimators': 9}
- Result: 99% accuracy on test data.
- Feature Importance Score shows the top 5 features are informative while the remaining are not.

	Features	Importance Score
0	satisfaction_level	0.382351
2	number_project	0.189290
4	time_spend_company	0.153951
3	average_monthly_hours	0.142197
1	last_evaluation	0.108173
5	Work_accident	0.007427
10	salary_medium	0.004956
9	salary_high	0.004895
7	managment	0.003182
6	promotion_last_5years	0.001980
8	RandD	0.001598

Logistic Regression

- kNN and RF were great in accurately predicting whether an employee has left, but does not help with remaining employees leaving in the future. LR can be the answer.

- Same features and scaling as kNN and use GridSearchCV. Result: 78% accuracy on test data (79% on cross validation)

- Tried with reduced features (top 5 from RF importance score) Result: 76% accuracy on test data. (76% is lower bound)

- Conclusion: Most likely there is nonlinearity in the features and feature interaction that the logistic regression model does not capture. Accuracy too low for usage.

K-Means Clustering

- Start with reduced features (top 5 from RF; scaled)
- Fit entire dataset into 7 clusters: below is the cluster centers and right is the proportion of employees that have left the company.
- Clusters 0, 1 and 3 have high probability of leaving. (vs. Avg. 24%)
  - 0: Happy and high performing bunch. Workload is in the higher end. Coming up on 5 years of service and possibly looking for different opportunities. Key group we need to focus on for retention.
  - 1: Poor performing group (bottom quartile). Not very happy. Does not have many projects assigned to and working hours are short.
  - 3: High performing, but overworked. Very disgruntled group. Need to reassign some of their projects to other groups (such as Clusters 6)

	left
cluster	
0	0.549252
1	0.594917
2	0.024390
3	0.582411
4	0.017674
5	0.018496
6	0.022891

	0	1	2	3	4	5	6
satisfaction_level	0.800510	0.418864	0.719596	0.157226	0.730666	0.632441	0.722155
time_spend_company	4.674549	3.059299	2.799443	4.255228	3.016859	7.647349	2.788147
number_project	4.560672	2.326146	3.675487	5.721402	4.148849	3.657213	3.522266
average_monthly_hours	244.263846	149.544474	242.528552	251.386839	160.866776	193.293465	190.363130
last_evaluation	0.901176	0.532892	0.636779	0.817146	0.613997	0.686178	0.884565

Conclusions

- kNN and RF were great in accurately predicting, but did not provide interpretation of the results, which could help determine which employees would leave next. (RF provided feature importance scores.)

- Logistic Regression produces coefficients and probabilities, but performed poorly in predicting past employees leaving.

- K-means with 7 clusters identified 3 clusters that have high employee attrition (55-60% vs avg. of 24%) and 4 clusters of low attrition (1-3%).

- Identify what type of employees each cluster represents then take necessary actions to improve on employee retention.

Recommendations and Next Steps

- Recommendations based on k-means clustering results
  Cluster 0 (Key focus): consider employees for promotion and new roles. They are satisfied, yet they are leaving to try different things.
  Cluster 3 (overworked): reassign some of their projects to other groups (such as Clusters 6 - high performing but not as busy)

- Find out why employees left and confirm if the clusters make sense.

- Try k-means clustering on only the employees that have left and compare against the three clusters from previous slide.

- Monitor performance of model as employee's features change, their "cluster" may also change.

Appendix

Feature statistics and correlation

	satisfaction_level	last_evaluation	number_project	average_monthly_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

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	Work_accident	left	promotion_last_5years
satisfaction_level	1.000000	0.105021	-0.142970	-0.020048	-0.100866	0.058697	-0.388375	0.025605
last_evaluation	0.105021	1.000000	0.349333	0.339742	0.131591	-0.007104	0.006567	-0.008684
number_project	-0.142970	0.349333	1.000000	0.417211	0.196786	-0.004741	0.023787	-0.006064
average_monthly_hours	-0.020048	0.339742	0.417211	1.000000	0.127755	-0.010143	0.071287	-0.003544
time_spend_company	-0.100866	0.131591	0.196786	0.127755	1.000000	0.002120	0.144822	0.067433
Work_accident	0.058697	-0.007104	-0.004741	-0.010143	0.002120	1.000000	-0.154622	0.039245
left	-0.388375	0.006567	0.023787	0.071287	0.144822	-0.154622	1.000000	-0.061788
promotion_last_5years	0.025605	-0.008684	-0.006064	-0.003544	0.067433	0.039245	-0.061788	1.000000

KNN code

```
# Scaling features
from sklearn.preprocessing import StandardScaler
stdsc = StandardScaler()
X_train_std = stdsc.fit_transform(X_train)
X_test_std = stdsc.transform(X_test)

# Cross validation
from sklearn.model_selection import ShuffleSplit
cv = ShuffleSplit(n_splits=20, test_size=0.3)

# kNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()

from sklearn.model_selection import GridSearchCV
parameters = {'n_neighbors': range(1,11), 'weights': ['uniform', 'distance']}
clf = GridSearchCV(knn, parameters, cv=cv)
clf.fit(X_train_std, y_train)
clf.best_params_
best_knn = clf.best_estimator_

best_knn.score(X_test_std, y_test)
```

RF code

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier()
rf_param = {'n_estimators': range(1,11)}
rf_grid = GridSearchCV(rf_model, rf_param, cv=cv)
rf_grid.fit(X_train, y_train)
rf_grid.best_params_
best_rf = rf_grid.best_estimator_

best_rf.score(X_test, y_test)

# feature importance scores
features = X.columns
feature_importances = best_rf.feature_importances_

features_df = pd.DataFrame({'Features': features, 'Importance Score': feature_importances})
features_df.sort_values('Importance Score', inplace=True, ascending=False)

features_df
```

K-means clustering

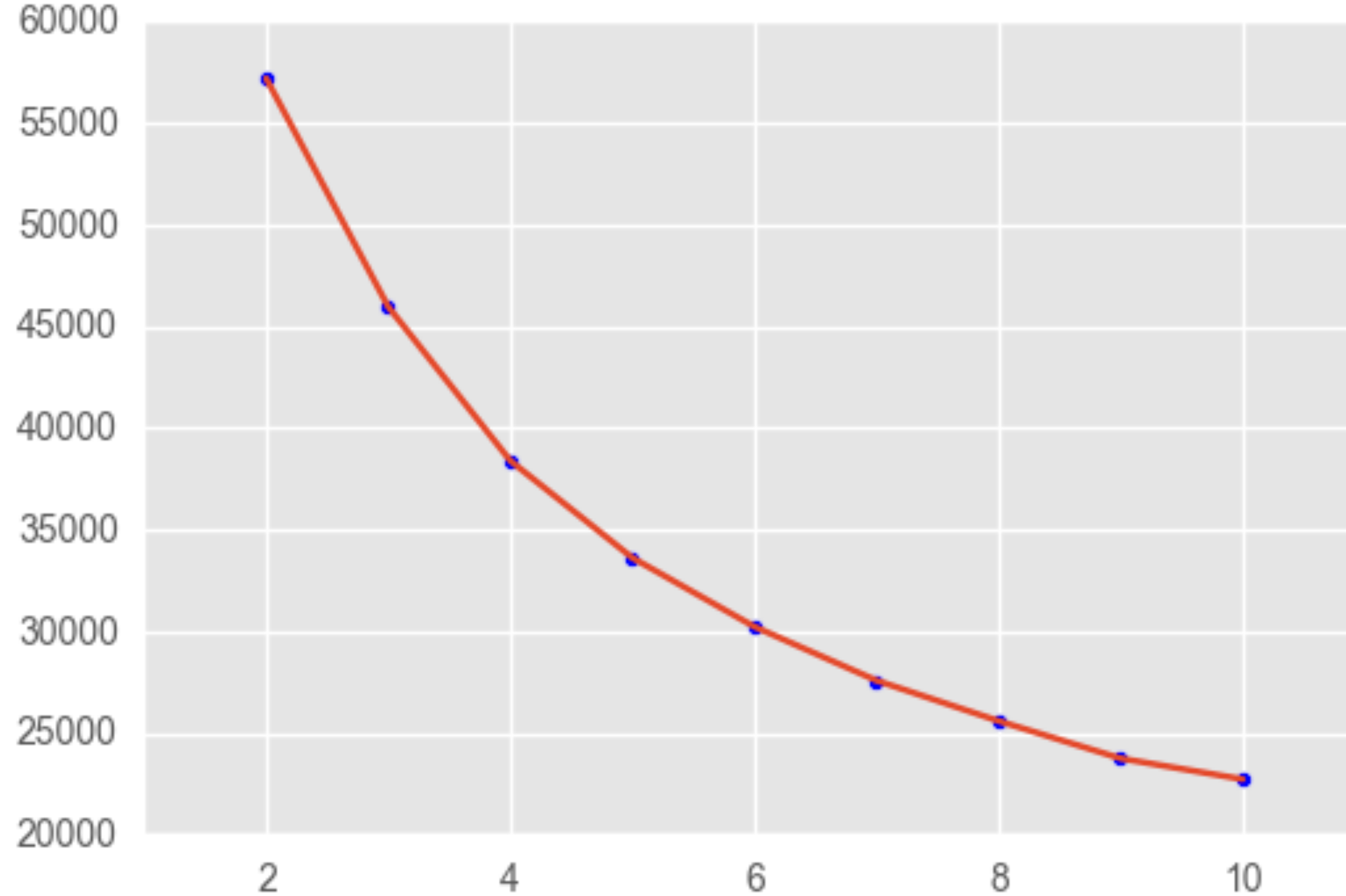
```
# K-mean clustering
from sklearn.cluster import KMeans

X2 = X[reduced_features]
X2_std = stdsc2.fit_transform(X2)

km = KMeans(n_clusters=7, n_init=20, random_state=7)
km.fit(X2_std)
columns = {str(x): stdsc2.inverse_transform(km.cluster_centers_[x])
            for x in range(0,len(km.cluster_centers_))}
pd.DataFrame(columns, index=X2.columns)

# Percentage of employees left for each cluster.
# Helps identify which cluster to direct our focus.
kmpredict = pd.DataFrame(data=df['left'])
kmpredict['cluster'] = km.labels_
kmpredict.groupby('cluster').mean()
```

K-means inertia graph




```
x = []
y = []
for n in range(2,11):
    km = KMeans(n_clusters=n, random_state=7)
    km.fit(X2_std)
    x.append(n)
    y.append(km.inertia_)
plt.scatter(x, y)
plt.plot(x,y);
```