Predicting IBM Employee Attrition Python Jupyter Notebook

Part 1 - Feature Engineering and Selection

A. Import Libraries and Data Set, and Inspect Data Set

Import numpy and pandas.

In [1]:
```
import numpy as np
import pandas as pd
```

Import data visualization libraries and set %matplotlib inline.

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Import IBM Employee Churn / Attrition comma-separated (CSV) file into a Pandas dataframe called churn.

In [3]:
```
churn = pd.read_csv('../data/ibm_hr_emp_churn.csv', sep=',')
```

Create copy of churn dataframe for exploratory data analysis and feature engineering.

In [4]:
```
churn1 = churn.copy()
```

View first five rows of churn dataframe.

In [5]:
```
churn1.head()
```

Out[5]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educati
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life S
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life S
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life S
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 35 columns

Obtain number of rows and columns in churn dataframe.

In [6]:
```
churn1.shape
```

Out[6]:
```
(1470, 35)
```

View structure of churn dataframe.

In [7]:
```
churn1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
Age                         1470 non-null int64
Attrition                   1470 non-null object
BusinessTravel              1470 non-null object
DailyRate                   1470 non-null int64
Department                  1470 non-null object
DistanceFromHome            1470 non-null int64
Education                   1470 non-null int64
EducationField              1470 non-null object
EmployeeCount               1470 non-null int64
EmployeeNumber              1470 non-null int64
EnvironmentSatisfaction     1470 non-null int64
Gender                      1470 non-null object
HourlyRate                  1470 non-null int64
JobInvolvement              1470 non-null int64
JobLevel                    1470 non-null int64
JobRole                     1470 non-null object
JobSatisfaction             1470 non-null int64
MaritalStatus               1470 non-null object
MonthlyIncome               1470 non-null int64
MonthlyRate                 1470 non-null int64
NumCompaniesWorked          1470 non-null int64
Over18                      1470 non-null object
OverTime                    1470 non-null object
PercentSalaryHike           1470 non-null int64
PerformanceRating           1470 non-null int64
RelationshipSatisfaction    1470 non-null int64
StandardHours               1470 non-null int64
StockOptionLevel            1470 non-null int64
TotalWorkingYears           1470 non-null int64
TrainingTimesLastYear       1470 non-null int64
WorkLifeBalance             1470 non-null int64
YearsAtCompany              1470 non-null int64
YearsInCurrentRole          1470 non-null int64
YearsSinceLastPromotion     1470 non-null int64
YearsWithCurrManager        1470 non-null int64
dtypes: int64(26), object(9)
memory usage: 402.0+ KB
```

Check for presence of missing values for all features.

In [8]:
```
churn1.isnull().sum()
```

Out[8]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

B. Explore and Engineer Categorical Features

Gather summary statistics for categorical features.

In [9]:
```
churn1.describe(include=['object'])
```

Out[9]:

	Attrition	BusinessTravel	Department	EducationField	Gender	JobRole	MaritalStatus	O
count	1470	1470	1470	1470	1470	1470	1470	
unique	2	3	3	6	2	9	3	
top	No	Travel_Rarely	Research & Development	Life Sciences	Male	Sales Executive	Married	
freq	1233	1043	961	606	882	326	673	

Obtain value counts for Attrition variable.

In [10]:
```
churn1.Attrition.value_counts()
```

Out[10]:
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

Generate Churn dummy variable by mapping Attrition categories to 0 or 1. (0 = No, 1 = Yes)

In [11]:
```
churn1['Churn'] = churn1.Attrition.map({'No':0, 'Yes':1})
churn1.Churn.value_counts()
```

Out[11]:
```
0    1233
1     237
Name: Churn, dtype: int64
```

Obtain value counts for BusinessTravel variable.

In [12]:
```
churn1.BusinessTravel.value_counts()
```

Out[12]:
```
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64
```

Convert BusinessTravel to numeric BusTravLevel (Business Travel Level) variable. (0 = Non-Travel, 1 = Travel_Rarely, 2 = Travel_Frequently)

In [13]:
```
churn1['BusTravLevel'] = churn1.BusinessTravel.map({'Travel_Rarely':1, 'Travel_Fr
churn1.BusTravLevel.value_counts()
```

Out[13]:
```
1    1043
2     277
0     150
Name: BusTravLevel, dtype: int64
```

Obtain value counts and employee churn probabilities for each Department.

In [14]:
```
churn1.groupby('Department').Churn.agg(['count', 'mean']).sort_values('mean', asc
```

Out[14]:

	count	mean
Department		
Sales	446	0.206278
Human Resources	63	0.190476
Research & Development	961	0.138398

Create Department dummy variables and add it to churn dataframe.

In [15]:
```
dept_dummies = pd.get_dummies(churn1.Department).drop('Research & Development', a
dept_dummies = dept_dummies.rename(columns={'Human Resources':'HR_Dept', 'Sales':
churn2 = pd.concat([churn1, dept_dummies], axis=1)
churn2.head()
```

Out[15]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educati
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life S
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life S
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life S
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 39 columns

Obtain value counts and employee churn probabilities for each Education Field.

In [16]:
```
churn2.groupby('EducationField').Churn.agg(['count', 'mean']).sort_values('mean',
```

Out[16]:

	count	mean
EducationField		
Human Resources	27	0.259259
Technical Degree	132	0.242424
Marketing	159	0.220126
Life Sciences	606	0.146865
Medical	464	0.135776
Other	82	0.134146

Create Education Field dummy variables and add it to churn dataframe.

In [17]:
```
edu_dummies = pd.get_dummies(churn2.EducationField).drop('Life Sciences', axis=1)
edu_dummies = edu_dummies.rename(columns={'Human Resources':'HR_Major', 'Technica
                                          'Market_Major', 'Medical':'Med_Major',
churn3 = pd.concat([churn2, edu_dummies], axis=1)
churn3.head()
```

Out[17]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educati
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life S
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life S
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life S
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 44 columns

Obtain value counts for Gender variable.

In [18]:
```
churn3.Gender.value_counts()
```

Out[18]:
```
Male      882
Female    588
Name: Gender, dtype: int64
```

Generate Gender_Dum dummy variable by mapping Gender categories to 0 or 1. (0 = Male, 1 = Female)

In [19]:
```
churn3['Gender_Dum'] = churn3.Gender.map({'Male':0, 'Female':1})
churn3.Gender_Dum.value_counts()
```

Out[19]:
```
0    882
1    588
Name: Gender_Dum, dtype: int64
```

Obtain value counts and employee churn probabilities for each Job Role.

In [20]:
```
churn3.groupby('JobRole').Churn.agg(['count', 'mean']).sort_values('mean', ascend
```

Out[20]:

JobRole	count	mean
Sales Representative	83	0.397590
Laboratory Technician	259	0.239382
Human Resources	52	0.230769
Sales Executive	326	0.174847
Research Scientist	292	0.160959
Manufacturing Director	145	0.068966
Healthcare Representative	131	0.068702
Manager	102	0.049020
Research Director	80	0.025000

Create Job Role dummy variables and add it to churn dataframe.

In [21]:
```
job_dummies = pd.get_dummies(churn3.JobRole).drop('Sales Executive', axis=1)
job_dummies = job_dummies.rename(columns={'Sales Representative':'Sales_Rep', 'La
                                          'Human Resources':'HR', 'Research Scien
                                          'Manufacturing Director':'Manuf_Dir', '
                                          'Manager':'Mgr', 'Research Director':'R
churn4 = pd.concat([churn3, job_dummies], axis=1)
churn4.head()
```

Out[21]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educati
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life S
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life S
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life S
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 53 columns

Obtain value counts and employee churn probabilities for each Marital Status.

In [22]:
```
churn4.groupby('MaritalStatus').Churn.agg(['count', 'mean']).sort_values('mean',
```

Out[22]:

MaritalStatus	count	mean
Single	470	0.255319
Married	673	0.124814
Divorced	327	0.100917

Create Marital Status dummy variables and add it to churn dataframe.

In [23]:
```
marital_dummies = pd.get_dummies(churn4.MaritalStatus).drop('Married', axis=1)
churn5 = pd.concat([churn4, marital_dummies], axis=1)
churn5.head()
```

Out[23]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educati
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life S
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life S
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life S
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 55 columns

Obtain value counts for Over18 variable.

In [24]:
```
churn5.Over18.value_counts()
```

Out[24]:
```
Y    1470
Name: Over18, dtype: int64
```

Obtain value counts for OverTime variable.

In [25]:
```
churn5.OverTime.value_counts()
```

Out[25]:
```
No     1054
Yes     416
Name: OverTime, dtype: int64
```

Generate Overtime_Dum dummy variable by mapping OverTime categories to 0 or 1. (0 = No, 1 = Yes)

In [26]:
```
churn5['Overtime_Dum'] = churn5.OverTime.map({'No':0, 'Yes':1})
churn5.Overtime_Dum.value_counts()
```

Out[26]:
```
0    1054
1     416
Name: Overtime_Dum, dtype: int64
```

Drop unengineered or unnecessary categorical features from churn dataframe.

In [27]:
```
churn_eng_cat = churn5.drop(['Attrition', 'BusinessTravel', 'Department', 'Educat
                             'Over18', 'OverTime'], axis=1)
```

Obtain number of rows and columns in churn dataframe with engineered categorical features and unengineered numerical features.

In [28]:
```
churn_eng_cat.shape
```

Out[28]:
```
(1470, 47)
```

View structure of churn dataframe with engineered categorical features and unengineered numerical features.

In [29]:
```
churn_eng_cat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 47 columns):
Age                         1470 non-null int64
DailyRate                   1470 non-null int64
DistanceFromHome            1470 non-null int64
Education                   1470 non-null int64
EmployeeCount               1470 non-null int64
EmployeeNumber              1470 non-null int64
EnvironmentSatisfaction     1470 non-null int64
HourlyRate                  1470 non-null int64
JobInvolvement              1470 non-null int64
JobLevel                    1470 non-null int64
JobSatisfaction             1470 non-null int64
MonthlyIncome               1470 non-null int64
MonthlyRate                 1470 non-null int64
NumCompaniesWorked          1470 non-null int64
PercentSalaryHike           1470 non-null int64
PerformanceRating           1470 non-null int64
RelationshipSatisfaction    1470 non-null int64
StandardHours               1470 non-null int64
StockOptionLevel            1470 non-null int64
TotalWorkingYears           1470 non-null int64
TrainingTimesLastYear       1470 non-null int64
WorkLifeBalance             1470 non-null int64
YearsAtCompany              1470 non-null int64
YearsInCurrentRole          1470 non-null int64
YearsSinceLastPromotion     1470 non-null int64
YearsWithCurrManager        1470 non-null int64
Churn                       1470 non-null int64
BusTravLevel                1470 non-null int64
HR_Dept                     1470 non-null uint8
Sales_Dept                  1470 non-null uint8
HR_Major                    1470 non-null uint8
Market_Major                1470 non-null uint8
Med_Major                   1470 non-null uint8
Other_Major                 1470 non-null uint8
Tech_Major                  1470 non-null uint8
Gender_Dum                  1470 non-null int64
HC_Rep                      1470 non-null uint8
HR                          1470 non-null uint8
Lab_Tech                    1470 non-null uint8
Mgr                         1470 non-null uint8
Manuf_Dir                   1470 non-null uint8
Research_Dir                1470 non-null uint8
Research_Sci                1470 non-null uint8
Sales_Rep                   1470 non-null uint8
Divorced                    1470 non-null uint8
Single                      1470 non-null uint8
Overtime_Dum                1470 non-null int64
dtypes: int64(30), uint8(17)
memory usage: 369.0 KB
```

C. Explore and Engineer Numerical Features

Drop unnecessary numerical features from churn dataframe.

In [30]:
```
churn6 = churn_eng_cat.drop(['EmployeeCount', 'EmployeeNumber', 'StandardHours'],
```

Remap ordered numerical features so that lowest level is 0 instead of 1.

In [31]:
```
churn6['Education'] = churn6.Education.map({1:0, 2:1, 3:2, 4:3, 5:4})
churn6['EnvironmentSatisfaction'] = churn6.EnvironmentSatisfaction.map({1:0, 2:1,
churn6['JobInvolvement'] = churn6.JobInvolvement.map({1:0, 2:1, 3:2, 4:3})
churn6['JobLevel'] = churn6.JobLevel.map({1:0, 2:1, 3:2, 4:3, 5:4})
churn6['JobSatisfaction'] = churn6.JobSatisfaction.map({1:0, 2:1, 3:2, 4:3})
churn6['PerformanceRating'] = churn6.PerformanceRating.map({1:0, 2:1, 3:2, 4:3})
churn6['RelationshipSatisfaction'] = churn6.RelationshipSatisfaction.map({1:0, 2:
churn6['WorkLifeBalance'] = churn6.WorkLifeBalance.map({1:0, 2:1, 3:2, 4:3})
```

Extract numerical features from churn dataframe to see correlation matrix between features.

In [32]:
```
num_features = ['Age', 'DailyRate', 'DistanceFromHome', 'Education', 'Environment
               'JobLevel', 'JobSatisfaction', 'MonthlyIncome', 'MonthlyRate', 'Nu
                'PerformanceRating', 'RelationshipSatisfaction', 'StockOptionLeve
               'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole', 'YearsS
churn_num_feat = churn6[num_features]
```

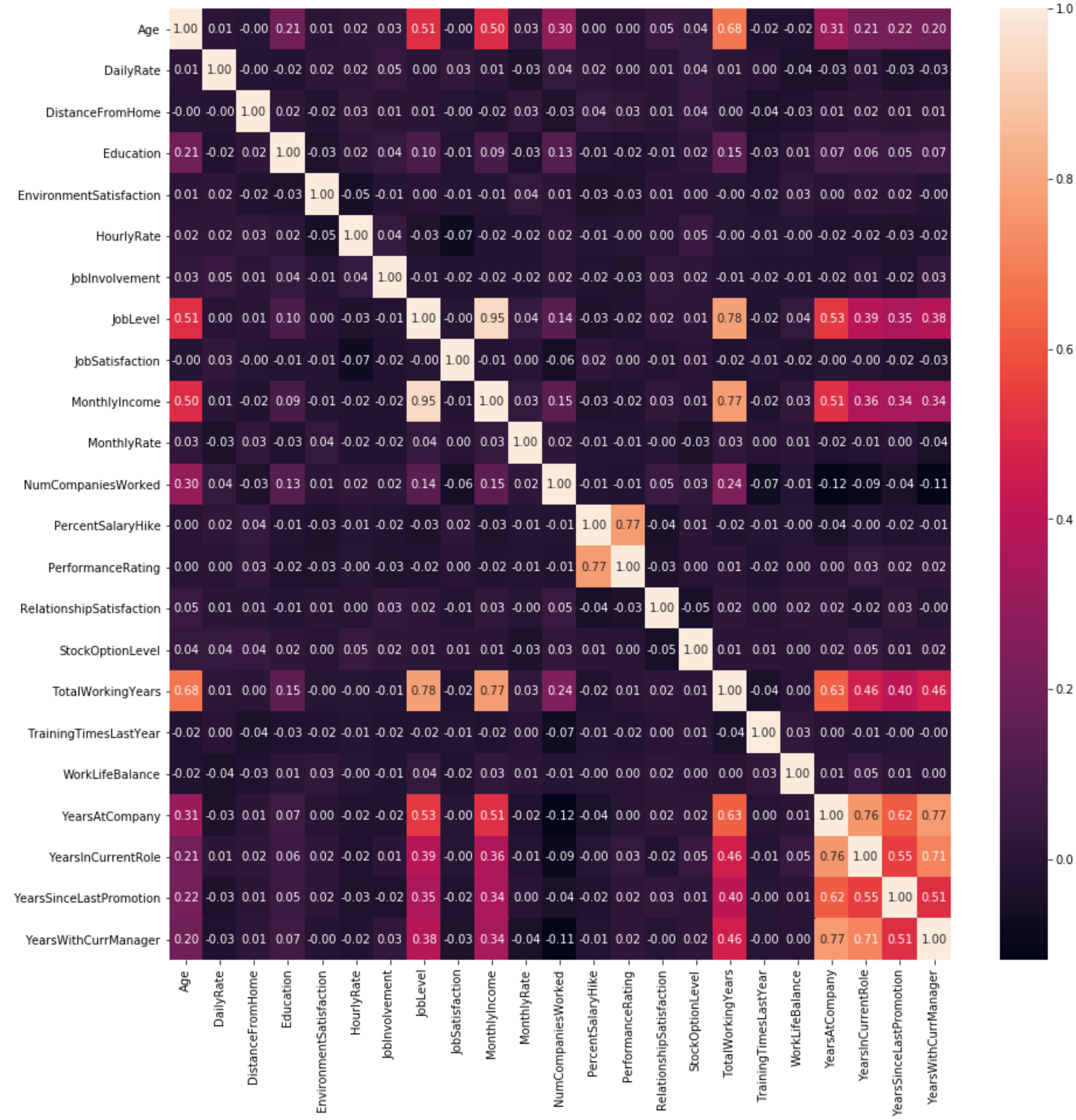
Check the number of numerical features.

In [33]:
```
churn_num_feat.shape
```

Out[33]:
```
(1470, 23)
```

View correlation matrix for numerical features.

In [34]:
```
plt.figure(figsize=(15, 15))
sns.heatmap(churn_num_feat.corr(), annot=True, fmt=".2f");
```



D. Feature Selection

Convert numerical feature data into numpy array and scale data to determine optimal number of features to include in predictive model.

In [35]:
```
from sklearn import decomposition
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
```

In [36]:
```
churn_num_feat_np = churn_num_feat.values
churn_num_feat_np_scaled = scale(churn_num_feat_np)
```

```
C:\Users\kyrma\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\util
s\validation.py:475: DataConversionWarning: Data with input dtype int64 was con
verted to float64 by the scale function.
  warnings.warn(msg, DataConversionWarning)
```

Create covariance matrix for 23 numerical features.

In [37]:
```
covar_matrix = PCA(n_components=23)
```

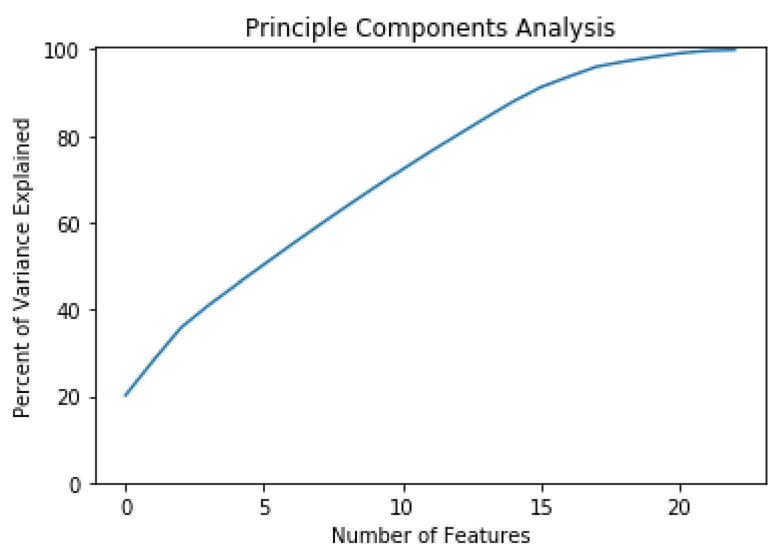
Calculate variance ratios.

In [38]:
```
covar_matrix.fit(churn_num_feat_np_scaled)
variance = covar_matrix.explained_variance_ratio_
var = np.cumsum(np.round(variance, decimals=3)*100)
var
```

Out[38]:
```
array([20.2, 28.2, 35.8, 41. , 45.7, 50.3, 54.9, 59.4, 63.8, 68.1, 72.3,
       76.4, 80.3, 84.2, 88. , 91.3, 93.7, 96. , 97.2, 98.2, 99.1, 99.7,
       99.9])
```

Determine the optimum number of features to include in predictive model.

In [39]:
```
plt.ylabel('Percent of Variance Explained')
plt.xlabel('Number of Features')
plt.title('Principle Components Analysis')
plt.ylim(0, 100.5)
plt.style.context('seaborn-whitegrid')
plt.plot(var);
```



- According to the principle components analysis graph, I should include ten features in my predictive model, which explains 68.1 percent of the variance. Therefore, I decided to include only ten features into my model. That way, I can minimize my model's bias and variance, reduce the risk of overfitting, and maximize model parsimony.

Define X and y for feature selection.

In [40]:
```
X = churn6.drop(['Churn'], axis=1)
y = churn6['Churn']
```

Select features by assessing their importance using random forest classifier method.

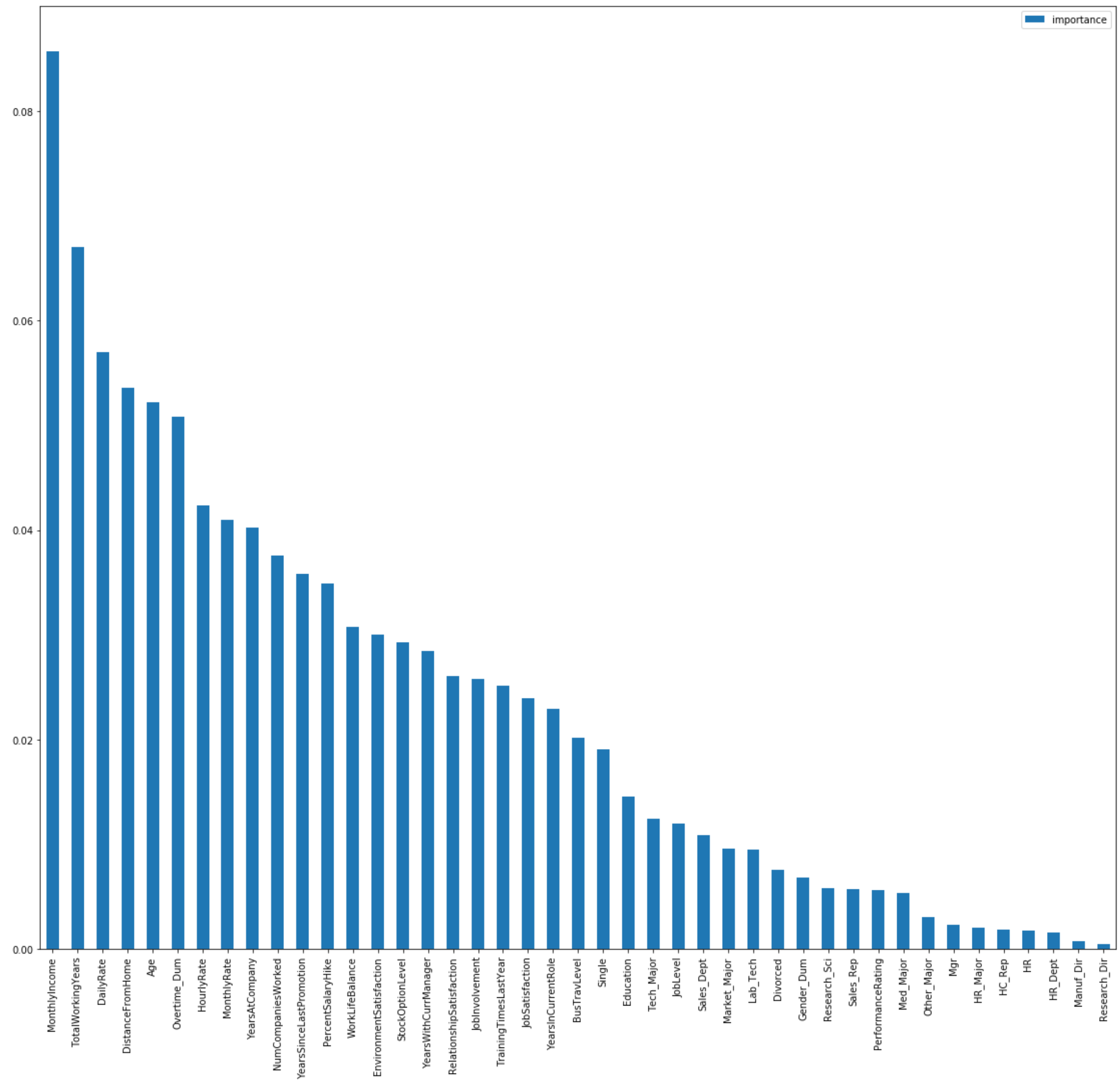
In [41]:
```
# Feature Selection: Embedded Method
from sklearn.ensemble import RandomForestClassifier
rfc_model = RandomForestClassifier(random_state=1)
rfc_model.fit(X, y)

rfc_feature_imp = pd.DataFrame(rfc_model.feature_importances_, index=X.columns, c
rfc_feat_imp_10 = rfc_feature_imp.sort_values('importance', ascending=False).head
rfc_feat_imp_10
```

```
C:\Users\kyrma\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ense
mble\weight_boosting.py:29: DeprecationWarning: numpy.core.umath_tests is an in
ternal NumPy module and should not be imported. It will be removed in a future
NumPy release.
  from numpy.core.umath_tests import inner1d
```

Out[41]:
```
Index(['MonthlyIncome', 'TotalWorkingYears', 'DailyRate', 'DistanceFromHome',
       'Age', 'Overtime_Dum', 'HourlyRate', 'MonthlyRate', 'YearsAtCompany',
       'NumCompaniesWorked'],
      dtype='object')
```

Plot random forest classifier method feature importances by descending order.

In [42]:
```
rfc_feature_imp.sort_values('importance', ascending=False).plot(kind='bar', figsi
```



Select features with filter method that removes all low-variance features.

In [43]:
```
# Feature Selection: Filter Method
from sklearn.feature_selection import VarianceThreshold, f_regression, SelectKBes

# Find all features with more than 90% variance in values.
threshold = 0.90
vt = VarianceThreshold().fit(X)

# Find feature names.
feat_var_threshold = X.columns[vt.variances_ > threshold * (1-threshold)]

# Select the top 10.
feat_var_threshold[0:10]
```

Out[43]:
```
Index(['Age', 'DailyRate', 'DistanceFromHome', 'Education',
       'EnvironmentSatisfaction', 'HourlyRate', 'JobInvolvement', 'JobLevel',
       'JobSatisfaction', 'MonthlyIncome'],
      dtype='object')
```

Filter for features selected by random forest classifier method but were not selected by filter method.

In [44]:
```
set(rfc_feat_imp_10) - set(feat_var_threshold[0:10])
```

Out[44]:
```
{'MonthlyRate',
 'NumCompaniesWorked',
 'Overtime_Dum',
 'TotalWorkingYears',
 'YearsAtCompany'}
```

Filter for features selected by filter method that removes all low-variance features but were not selected by random forest classifier method.

In [45]:
```
set(feat_var_threshold[0:10]) - set(rfc_feat_imp_10)
```

Out[45]:
```
{'Education',
 'EnvironmentSatisfaction',
 'JobInvolvement',
 'JobLevel',
 'JobSatisfaction'}
```

Select features based on univariate statistical tests.

In [46]:
```
# Feature Selection: Filter Method
X_scored = SelectKBest(score_func=f_regression, k='all').fit(X, y)
feature_scoring = pd.DataFrame({'feature': X.columns, 'score': X_scored.scores_})

feat_scored_10 = feature_scoring.sort_values('score', ascending=False).head(10)['
feat_scored_10
```

Out[46]:
```
array(['Overtime_Dum', 'Single', 'TotalWorkingYears', 'JobLevel',
       'YearsInCurrentRole', 'MonthlyIncome', 'Age', 'Sales_Rep',
       'YearsWithCurrManager', 'StockOptionLevel'], dtype=object)
```

Select features by eliminating them recursively via wrapper method.

In [47]:
```
# Feature Selection: Wrapper Method
from sklearn.linear_model import LogisticRegression

# Select 10 features by using recursive feature elimination (RFE) with logistic re
from sklearn.feature_selection import RFE
rfe = RFE(LogisticRegression(), 10)
rfe.fit(X, y)

feature_rfe_scoring = pd.DataFrame({'feature': X.columns, 'score': rfe.ranking_})

feat_rfe_10 = feature_rfe_scoring[feature_rfe_scoring['score'] == 1]['feature'].v
feat_rfe_10
```

Out[47]:
```
array(['JobInvolvement', 'BusTravLevel', 'HR_Major', 'Tech_Major', 'HR',
       'Lab_Tech', 'Research_Dir', 'Sales_Rep', 'Single', 'Overtime_Dum'],
      dtype=object)
```

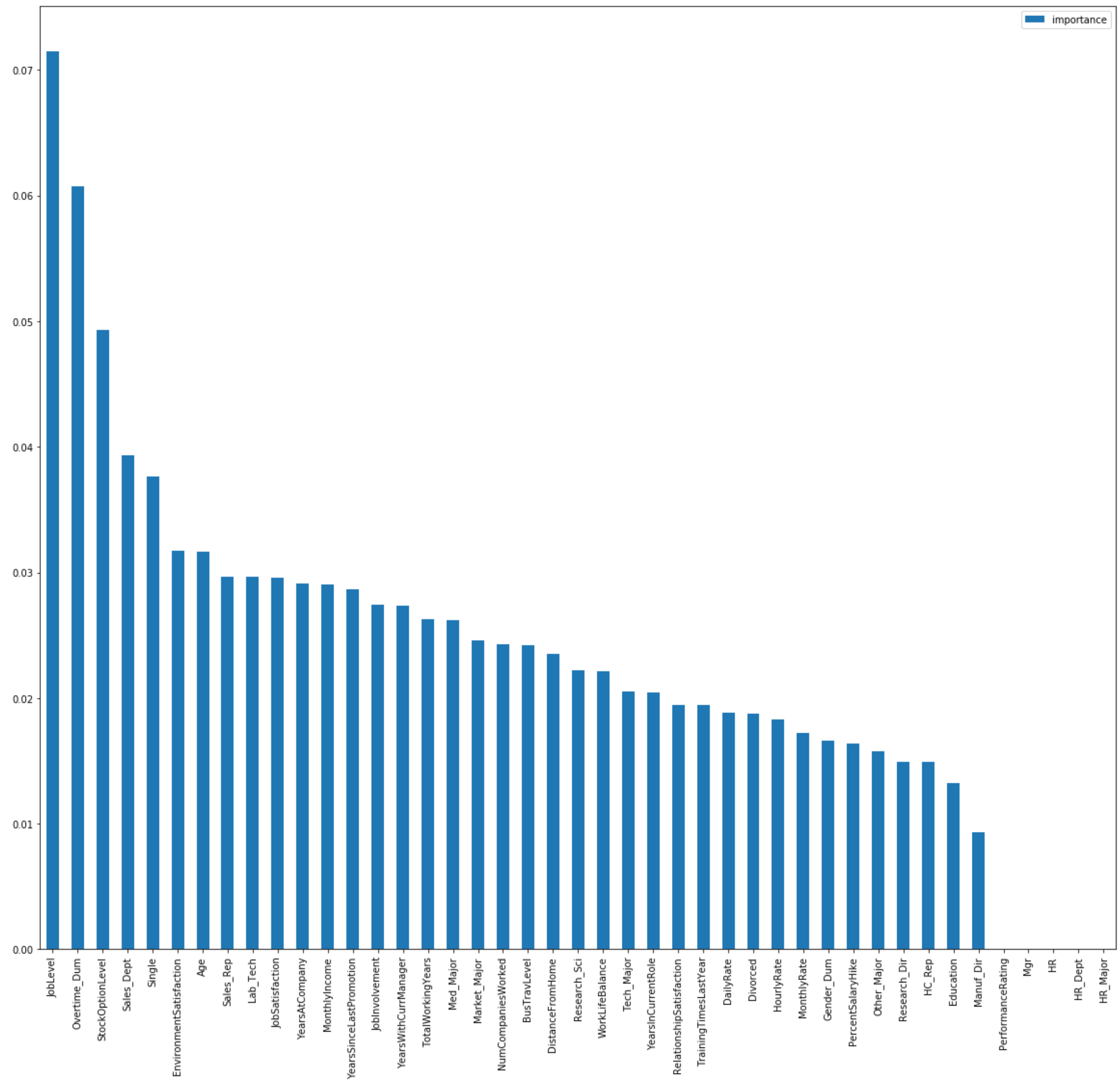
Select features by assessing their importance using XGBoost classifier method.

In [48]:
```
import xgboost as xgb
xgb_model = xgb.XGBClassifier(n_estimators=500, random_state=1)
xgb_model.fit(X, y)
xgb_feature_imp = pd.DataFrame(xgb_model.feature_importances_, index=X.columns, co
xgb_feat_imp_10 = xgb_feature_imp.sort_values('importance', ascending=False).head
xgb_feat_imp_10
```

Out[48]:
```
Index(['JobLevel', 'Overtime_Dum', 'StockOptionLevel', 'Sales_Dept', 'Single',
       'EnvironmentSatisfaction', 'Age', 'Sales_Rep', 'Lab_Tech',
       'JobSatisfaction'],
      dtype='object')
```

Plot XGBoost classifier feature importances by descending order.

In [49]:
```
xgb_feature_imp.sort_values('importance', ascending=False).plot(kind='bar', figsi
```



Gather unique features from all five feature selection methods.

In [50]:
```
features = np.hstack([feat_var_threshold[0:10], rfc_feat_imp_10, feat_scored_10, 

features = np.unique(features)
print('Final features set:\n')
for f in features:
    print("\t-{}".format(f))
```

```
Final features set:

	-Age
	-BusTravLevel
	-DailyRate
	-DistanceFromHome
	-Education
	-EnvironmentSatisfaction
	-HR
	-HR_Major
	-HourlyRate
	-JobInvolvement
	-JobLevel
	-JobSatisfaction
	-Lab_Tech
	-MonthlyIncome
	-MonthlyRate
	-NumCompaniesWorked
	-Overtime_Dum
	-Research_Dir
	-Sales_Dept
	-Sales_Rep
	-Single
	-StockOptionLevel
	-Tech_Major
	-TotalWorkingYears
	-YearsAtCompany
	-YearsInCurrentRole
	-YearsWithCurrManager
```

- Based off the above unique and selected features from these five feature selection methods, numerical feature correlation matrix heatmap, and business logic / domain knowledge, I have decided to include only these ten features to build the machine learning models: Age, BusTravLevel, DistanceFromHome, EnvironmentSatisfaction, JobInvolvement, MonthlyIncome, Overtime_Dum, Sales_Rep, Single, and StockOptionLevel.

Create churn / attrition modeling data by selecting target feature and predictor features for modeling.

In [51]:
```
modeling_cols = ['Churn', 'Age', 'BusTravLevel', 'DistanceFromHome', 'Environment
                 'MonthlyIncome', 'Overtime_Dum', 'Sales_Rep', 'Single', 'StockOp
churn_model = churn6[modeling_cols]
```

Obtain value counts and employee churn probabilities for Overtime_Dum (Overtime dummy) variable, a categorical feature that highly impacts likelihood of employee to churn.

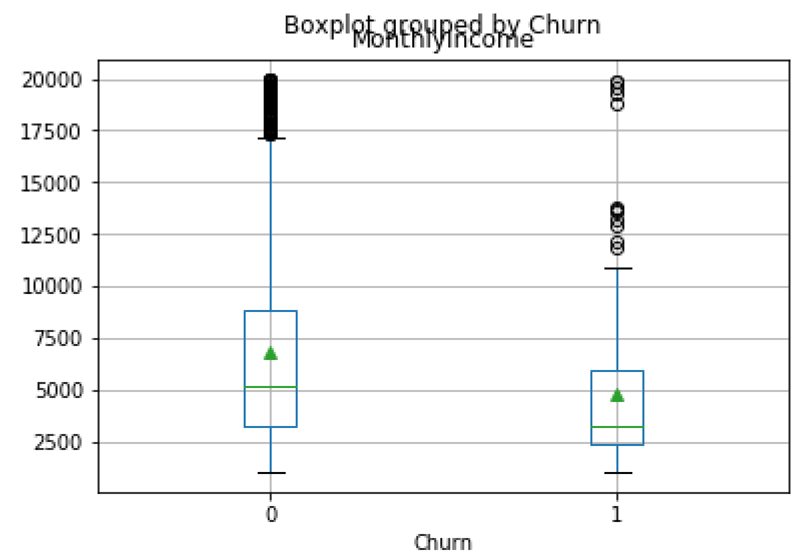
In [52]:
```
churn_model.groupby('Overtime_Dum').Churn.agg(['count', 'mean']).sort_values('mea
```

Out[52]:

	count	mean
Overtime_Dum		
1	416	0.305288
0	1054	0.104364

Generate histogram for Monthly Income, a numerical feature that highly impacts likelihood of employee to churn.

In [53]:
```
churn_model.boxplot(column='MonthlyIncome', by='Churn', showmeans=True);
```



Export finalized churn modeling dataframe to CSV file.

In [54]:
```
churn_model.to_csv('../data/churn_modeling_data.csv', sep=',', index=False)
```

Save finalized churn modeling dataframe to pickle file for subsequent classification model notebooks.

In [55]:
```
churn_model.to_pickle('../data/churn_modeling_data.pickle')
```