In [2]:

**import** **numpy** **as** **np**  
**import** **pandas** **as** **pd**  
**import** **matplotlib.pyplot** **as** **plt**  
**import** **seaborn** **as** **sns**  
%**matplotlib** inline

In [3]:

data = pd.read\_csv('E:/Test/Train.csv')

In [4]:

data.head()

Out[4]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Age** | **Attrition** | **BusinessTravel** | **DailyRate** | **Department** | **DistanceFromHome** | **Education** | **EducationField** | **EmployeeCount** | **EmployeeNumber** | **...** | **RelationshipSatisfaction** | **StandardHours** | **StockOptionLevel** | **TotalWorkingYears** | **TrainingTimesLastYear** | **WorkLifeBalance** | **YearsAtCompany** | **YearsInCurrentRole** | **YearsSinceLastPromotion** | **YearsWithCurrManager** |
| **0** | 41 | Yes | Travel\_Rarely | 1102 | Sales | 1 | 2 | Life Sciences | 1 | 1 | ... | 1 | 80 | 0 | 8 | 0 | 1 | 6 | 4 | 0 | 5 |
| **1** | 49 | No | Travel\_Frequently | 279 | Research & Development | 8 | 1 | Life Sciences | 1 | 2 | ... | 4 | 80 | 1 | 10 | 3 | 3 | 10 | 7 | 1 | 7 |
| **2** | 37 | Yes | Travel\_Rarely | 1373 | Research & Development | 2 | 2 | Other | 1 | 4 | ... | 2 | 80 | 0 | 7 | 3 | 3 | 0 | 0 | 0 | 0 |
| **3** | 33 | No | Travel\_Frequently | 1392 | Research & Development | 3 | 4 | Life Sciences | 1 | 5 | ... | 3 | 80 | 0 | 8 | 3 | 3 | 8 | 7 | 3 | 0 |
| **4** | 27 | No | Travel\_Rarely | 591 | Research & Development | 2 | 1 | Medical | 1 | 7 | ... | 4 | 80 | 1 | 6 | 3 | 3 | 2 | 2 | 2 | 2 |

5 rows × 35 columns

The above data consists of a dependent variable Attrition and others as independent variables. Exploring the data would provide explain which model to use.

**Data Exploration**[**¶**](#gjdgxs)

In [5]:

plt.figure(figsize=(12,7))  
sns.heatmap(data.isnull(),yticklabels=**False**,cbar=**False**,cmap='viridis')  
plt.show()

In the above Chart we checked for missing values using a heatmap. Thus from the above visualisation we can say that there are no NA in this data.

In [6]:

plt.figure(figsize=(14,10))  
sns.heatmap(data.corr(),yticklabels=**False**,cbar=**True**,linewidths=0)  
plt.show()

The above data shows there is a huge serial correlation in the data. Dropping these features may cause to lose enough information. We will try using decomposition to solve this problem.

In [7]:

plt.figure(figsize=(8,8))  
sns.jointplot(data['Age'],data['DailyRate'])  
plt.show()

<matplotlib.figure.Figure at 0x81456e0630>

The above shows the joint plot of Kernel Density distribution of Daily Rate against Age.

In [8]:

plt.figure(figsize=(8,8))  
sns.barplot(x=data['Department'],y=data['DailyRate'],hue=data['EducationField'])  
plt.show()

E:\Python\Anaconda\lib\site-packages\seaborn\categorical.py:1508: FutureWarning: remove\_na is deprecated and is a private function. Do not use.  
 stat\_data = remove\_na(group\_data[hue\_mask])

In [9]:

sns.countplot(data['Attrition'])  
plt.show()

E:\Python\Anaconda\lib\site-packages\seaborn\categorical.py:1460: FutureWarning: remove\_na is deprecated and is a private function. Do not use.  
 stat\_data = remove\_na(group\_data)

In [10]:

data.columns

Out[10]:

Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',  
 'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',  
 'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',  
 'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',  
 'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',  
 'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',  
 'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',  
 'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',  
 'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',  
 'YearsWithCurrManager'],  
 dtype='object')

**Feature Engineering**[**¶**](#30j0zll)

In [11]:

data.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

**From the above, we can see that there are 9 categorical data. Here we have to create dummy variables of them.**

In [12]:

BusinessTravel = pd.get\_dummies(data['BusinessTravel'],drop\_first=**True**)

In [13]:

Department = pd.get\_dummies(data['Department'],drop\_first=**True**)

In [14]:

EducationField = pd.get\_dummies(data['EducationField'],drop\_first=**True**)

In [15]:

Gender = pd.get\_dummies(data['Gender'],drop\_first=**True**)

In [16]:

JobRole = pd.get\_dummies(data['JobRole'],drop\_first=**True**)

In [17]:

MaritalStatus = pd.get\_dummies(data['MaritalStatus'],drop\_first=**True**)

In [18]:

Train = data

In [19]:

**def** StrToBin(a):  
 **if** a == 'Yes':  
 **return** 1  
 **else**:  
 **return** 0

In [20]:

**def** StrToBinb(a):  
 **if** a == 'Y':  
 **return** 1  
 **else**:  
 **return** 0

In [21]:

Train['Attrition']=Train['Attrition'].apply(StrToBin)

In [22]:

Train['OverTime']=Train['OverTime'].apply(StrToBin)

In [23]:

Train['Over18']=Train['Over18'].apply(StrToBinb)

In [24]:

Train.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 int64  
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 int64  
OverTime 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  
dtypes: int64(29), object(6)  
memory usage: 402.0+ KB

In [25]:

Train.drop(['Department','EducationField','Gender','BusinessTravel','JobRole','MaritalStatus'],axis=1,inplace=**True**)

In [26]:

Train = pd.concat([Train,Department,EducationField,Gender,BusinessTravel,JobRole,MaritalStatus],axis=1)

In [27]:

Train.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1470 entries, 0 to 1469  
Data columns (total 49 columns):  
Age 1470 non-null int64  
Attrition 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  
Over18 1470 non-null int64  
OverTime 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  
Research & Development 1470 non-null uint8  
Sales 1470 non-null uint8  
Life Sciences 1470 non-null uint8  
Marketing 1470 non-null uint8  
Medical 1470 non-null uint8  
Other 1470 non-null uint8  
Technical Degree 1470 non-null uint8  
Male 1470 non-null uint8  
Travel\_Frequently 1470 non-null uint8  
Travel\_Rarely 1470 non-null uint8  
Human Resources 1470 non-null uint8  
Laboratory Technician 1470 non-null uint8  
Manager 1470 non-null uint8  
Manufacturing Director 1470 non-null uint8  
Research Director 1470 non-null uint8  
Research Scientist 1470 non-null uint8  
Sales Executive 1470 non-null uint8  
Sales Representative 1470 non-null uint8  
Married 1470 non-null uint8  
Single 1470 non-null uint8  
dtypes: int64(29), uint8(20)  
memory usage: 361.8 KB

As we can see, there are many features in this data. Removing the features might lose information. Instead of feature selection, We are going to extract features from this data using Linear Discriminant Analysis.

In [28]:

m = list(Train.columns)  
n = list(filter(**lambda** t: t **not** **in** ['Attrition'], m))

In [29]:

X = Train[n]

In [30]:

y = Train['Attrition']

**Train Test Split**[**¶**](#1fob9te)

In [31]:

**from** **sklearn.model\_selection** **import** train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state = 0)

In [32]:

**from** **sklearn.preprocessing** **import** StandardScaler  
sc = StandardScaler()  
X\_train = sc.fit\_transform(X\_train)  
X\_test = sc.transform(X\_test)

**MODEL 1: Logistic Regression with LDA**[**¶**](#3znysh7)

**Applying LDA**[**¶**](#2et92p0)

In [33]:

**from** **sklearn.discriminant\_analysis** **import** LinearDiscriminantAnalysis **as** LDA

In [34]:

lda = LDA(n\_components = 2)  
X\_train = lda.fit\_transform(X\_train, y\_train)  
X\_test = lda.transform(X\_test)

E:\Python\Anaconda\lib\site-packages\sklearn\discriminant\_analysis.py:388: UserWarning: Variables are collinear.  
 warnings.warn("Variables are collinear.")

**Logistic Regression**[**¶**](#tyjcwt)

In [35]:

**from** **sklearn.linear\_model** **import** LogisticRegression  
classifier = LogisticRegression(penalty='l2', solver='sag', C=1)  
fit1 = classifier.fit(X\_train, y\_train)

In [36]:

y\_pred = fit1.predict(X\_test)

**Confusion Matrix and Classification Report**[**¶**](#3dy6vkm)

In [37]:

**from** **sklearn.metrics** **import** confusion\_matrix,classification\_report  
cm = confusion\_matrix(y\_test, y\_pred)  
cr = classification\_report(y\_test, y\_pred)

In [38]:

print(cm)  
print(cr)

[[359 12]  
 [ 41 29]]  
 precision recall f1-score support  
  
 0 0.90 0.97 0.93 371  
 1 0.71 0.41 0.52 70  
  
avg / total 0.87 0.88 0.87 441

**Applying LDA and then performing Logistic Regression gives us an accuracy score of 87%.**

**MODEL 2: K- Nearest Neighbour**[**¶**](#1t3h5sf)

**Train Test Split**[**¶**](#1fob9te)

**In [39]:**

**from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)**

**Applying LDA**

**In [40]:**

**from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA**

**In [41]:**

**lda = LDA(n\_components = 2,solver='eigen',shrinkage='auto')  
X\_train = lda.fit\_transform(X\_train, y\_train)  
X\_test = lda.transform(X\_test)**

**E:\Python\Anaconda\lib\site-packages\sklearn\utils\validation.py:444: DataConversionWarning: Data with input dtype int64 was converted to float64 by StandardScaler.  
 warnings.warn(msg, DataConversionWarning)**

**1) Performing KNN**

**In [42]:**

**from sklearn.neighbors import KNeighborsClassifier  
knn = KNeighborsClassifier(n\_neighbors=17,weights='distance',algorithm='brute')  
knn.fit(X\_train,y\_train)**

**Out[42]:**

**KNeighborsClassifier(algorithm='brute', leaf\_size=30, metric='minkowski',  
 metric\_params=None, n\_jobs=1, n\_neighbors=17, p=2,  
 weights='distance')**

**In [43]:**

**pred = knn.predict(X\_test)**

**Choosing the best K**

**In [44]:**

**error\_rate = []  
for i in range(1,40):  
 knn = KNeighborsClassifier(n\_neighbors=i)  
 knn.fit(X\_train,y\_train)  
 pred\_i = knn.predict(X\_test)  
 error\_rate.append(np.mean(pred\_i != y\_test))**

**In [45]:**

**plt.figure(figsize=(10,6))  
plt.plot(range(1,40),error\_rate,color='blue', linestyle='dashed', marker='o',  
 markerfacecolor='red', markersize=10)  
plt.title('Error Rate vs. K Value')  
plt.xlabel('K')  
plt.ylabel('Error Rate')**

**Out[45]:**

**Text(0,0.5,'Error Rate')**

**Report**

**In [46]:**

**from sklearn.metrics import classification\_report,confusion\_matrix  
print(confusion\_matrix(y\_test,pred))  
print(classification\_report(y\_test,pred))**

**[[228 17]  
 [ 27 22]]  
 precision recall f1-score support  
  
 0 0.89 0.93 0.91 245  
 1 0.56 0.45 0.50 49  
  
avg / total 0.84 0.85 0.84 294**

**MODEL 3: SVM**[**¶**](#4d34og8)

**Train Test Split**[**¶**](#1fob9te)

**In [47]:**

**from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state = 0)**

**Applying LDA**

**In [48]:**

**from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA  
lda = LDA(n\_components = 2,solver='svd')  
X\_train = lda.fit\_transform(X\_train, y\_train)  
X\_test = lda.transform(X\_test)**

**E:\Python\Anaconda\lib\site-packages\sklearn\discriminant\_analysis.py:388: UserWarning: Variables are collinear.  
 warnings.warn("Variables are collinear.")**

**In [49]:**

**from sklearn.svm import SVC  
model = SVC(C=100,kernel = 'rbf')  
model.fit(X\_train,y\_train)**

**Out[49]:**

**SVC(C=100, cache\_size=200, class\_weight=None, coef0=0.0,  
 decision\_function\_shape='ovr', degree=3, gamma='auto', kernel='rbf',  
 max\_iter=-1, probability=False, random\_state=None, shrinking=True,  
 tol=0.001, verbose=False)**

**In [50]:**

**predictions = model.predict(X\_test)**

**Report**

**In [51]:**

**from sklearn.metrics import classification\_report,confusion\_matrix  
print(confusion\_matrix(y\_test,predictions))  
print(classification\_report(y\_test,predictions))**

**[[359 12]  
 [ 41 29]]  
 precision recall f1-score support  
  
 0 0.90 0.97 0.93 371  
 1 0.71 0.41 0.52 70  
  
avg / total 0.87 0.88 0.87 441**

**Decision Tree**[**¶**](#2s8eyo1)

**In [52]:**

**from sklearn.tree import DecisionTreeClassifier as DTC**

**In [53]:**

**dt = DTC(criterion='entropy',splitter='random')**

**In [54]:**

**tree = dt.fit(X\_train,y\_train)**

**In [55]:**

**pred\_t = dt.predict(X\_test)**

**In [56]:**

**cm = confusion\_matrix(y\_test,pred\_t)**

**In [57]:**

**cr = classification\_report(y\_test,pred\_t)**

**In [58]:**

**print(cm)  
print(cr)**

**[[337 34]  
 [ 41 29]]  
 precision recall f1-score support  
  
 0 0.89 0.91 0.90 371  
 1 0.46 0.41 0.44 70  
  
avg / total 0.82 0.83 0.83 441**

**K-FOLD CROSS VALIDATION**[**¶**](#17dp8vu)

**Logistic Regression**

**In [59]:**

**from sklearn.model\_selection import cross\_val\_score  
accuracies = cross\_val\_score(estimator = fit1, X = X\_train, y = y\_train, cv = 10)  
print('mean: ' ,accuracies.mean())  
print('SD:' ,accuracies.std())  
print(accuracies)**

**mean: 0.885384542166  
SD: 0.0179605849583  
[ 0.875 0.875 0.89320388 0.85436893 0.86407767 0.91262136  
 0.88349515 0.89215686 0.91176471 0.89215686]**

**KNN**

**In [60]:**

**from sklearn.model\_selection import cross\_val\_score  
accuracies = cross\_val\_score(estimator = knn, X = X\_train, y = y\_train, cv = 10)  
print('mean: ' ,accuracies.mean())  
print('SD:' ,accuracies.std())  
print(accuracies)**

**mean: 0.888306681896  
SD: 0.0210063164393  
[ 0.85576923 0.89423077 0.88349515 0.87378641 0.85436893 0.9223301  
 0.89320388 0.89215686 0.91176471 0.90196078]**

**SVM**

**In [61]:**

**from sklearn.model\_selection import cross\_val\_score  
accuracies = cross\_val\_score(estimator = model, X = X\_train, y = y\_train, cv = 10)  
print('mean: ' ,accuracies.mean())  
print('SD:' ,accuracies.std())  
print(accuracies)**

**mean: 0.882452884066  
SD: 0.0178607117406  
[ 0.875 0.875 0.89320388 0.85436893 0.85436893 0.91262136  
 0.88349515 0.88235294 0.90196078 0.89215686]**

**Decision Tree**

**In [62]:**

**from sklearn.model\_selection import cross\_val\_score  
accuracies = cross\_val\_score(estimator = tree, X = X\_train, y = y\_train, cv = 10)  
print('mean: ' ,accuracies.mean())  
print('SD:' ,accuracies.std())  
print(accuracies)**

**mean: 0.821209308966  
SD: 0.0407573073695  
[ 0.76923077 0.85576923 0.77669903 0.83495146 0.83495146 0.80582524  
 0.86407767 0.88235294 0.83333333 0.75490196]**

**From this we can see that the best algorithm to predict if an employee will leave or not is KNN followed by Logistic Regression.**

**Thank You**[**¶**](#3rdcrjn)