ISL Assignment 2 -Q1

1.a. Consider first the subset that consists only of Rain and Snow. There are 226 entries with these two categories.

i. Apply logistic regression, LDA, QDA, and knn on this dataset to determine the accuracy, precision, and recall of these models. You're to use randomly 180 days for the training set (approximately 80% of 226) and the rest for the test data set. Conduct your study over 100 replications, and summary the result of your analysis with your conclusion which models you'll recommend using based on the metrics: accuracy, precision and recall.

Solution:

For sub setting the data from the actual data set KC weather below command is executed in the R. (actual data consists of 366 entries by sub setting for the Events -Rain and Snow is 226 entries)

```
loadhistory("~/R/kc_weather.csv")
> data<-kc_weather
> View(data)
>datasubset<-subset(data, !(Events %in% c("Rain_Thunderstorm")))
```

Splitting the dataset for the training set as 80% and the rest for the testing data set.

```
library("caTools", lib.loc="~/R/win-library/3.4")
> library(caTools)
> split<-sample.split(datasubset,SplitRatio = 0.8)
> split
[1]  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
> training<-subset(datasubset,split=="TRUE")
> testing<-subset(datasubset,split=="FALSE")
> TotalEntries=226
> Training=176 (80 % of total entries)
> Testing=TotalEntries-Training
> rep=100
>accuracy=dim(rep)
>precision=dim(rep)
>recall=dim(rep)
```

Logistic Regression on 226 entries with 100 replications:

Logistic Regression, or logit regression is a regression model where the dependent variable is categorical. The binary logistic model is used to estimate the probability of a binary response based on one or more predictor.

```
for(k in 1:rep){
    train=sample(1:TotalEntries,Training)
    datasubset.train= datasubset[train, 2:9]
    datasubset.test=datasubset[-train,2:9]

model=glm(Events~Temp.F+Dew_Point.F+Humidity.percentage+Sea_Level_Press.in+Visibility.mi+Wind.mph+Precip.in
,datasubset.train,family="binomial")
    res=predict(model,datasubset.test)
    tablin=table(Actualvalue=datasubset.test$Events,Predictedvalue=res>0.5)
    errlin[k]=(Testing-sum(diag(tablin)))/Testing
    accuracy[k]=sum(diag(tablin))/Testing
     precision[k]=tablin[1,1]/sum(tablin[1,1:2])
    recall[k]=tablin[1,1]/sum(tablin[1:2,1])
}

> meanaccuracy
[1] 0.9514
> meanprecision=mean(precision)
> meanprecision
[1] 0.9670328
> meanrecall=mean(recall)
> meanrecall
[1] 0.9712895
```

LDA on 226 entries with 100 replications:

Linear discriminant analysis (assumes as normal distribution and applied with classes as more than 2) is the statistical method of classifying an observation having p component in one of the two groups. It gives two regions separated by a line so that helps in classifying the given data. The region & separate line is the defined by linear discriminant function.

```
for(k in 1:rep){
+     train=sample(1:TotalEntries,Training)
+
datasubset.lda_train=lda(Events~Temp.F+Dew_Point.F+Humidity.percentage+Sea_Level_Press.in+Visibility.mi+Win
d.mph+Precip.in,datasubset[train,])
+     predict(datasubset.lda_train,datasubset[-train,])$class
+     tablin=table(datasubset$Events[-train],predict(datasubset.lda_train,datasubset[-train,])$class)
+     errlin[k]=(Testing-sum(diag(tablin)))/Testing
+     accuracy[k]=sum(diag(tablin))/Testing
+     precision[k]=tablin[1,1]/sum(tablin[1,1:2])
+     recall[k]=tablin[1,1]/sum(tablin[1:2,1])
+ }
> meanaccuracy=mean(accuracy)
> meanaccuracy
[1] 0.9312
> meanprecision=mean(precision)
> meanprecision
[1] 0.9474137
> meanrecall=mean(recall)
> meanrecall
[1] 0.9645754
```

QDA on 226 entries with 100 replications:

Quadratic discriminant analysis, is modelled as a multivariate Gaussian distribution with density: In the case of QDA, there are no assumptions on the covariance matrices of the Gaussians, leading to **quadratic** decision surfaces.

```
for(k in 1:rep){
+     train=sample(1:TotalEntries,Training)
+
datasubset.qda_train=qda(Events~Temp.F+Dew_Point.F+Humidity.percentage+Sea_Level_Press.in+Visibility.mi+Win
d.mph+Precip.in,datasubset[train,])
+     predict(datasubset.qda_train,datasubset[-train,])$class
+     tablin=table(datasubset$Events[-train],predict(datasubset.qda_train,datasubset[-train,])$class)
+     errlin[k]=(Testing-sum(diag(tablin)))/Testing
+     accuracy[k]=sum(diag(tablin))/Testing
+     precision[k]=tablin[1,1]/sum(tablin[1,1:2])
+     recall[k]=tablin[1,1]/sum(tablin[1:2,1])
+ }
> meanaccuracy=mean(accuracy)
> meanaccuracy
[1] 0.9354
> meanprecision=mean(precision)
> meanprecision
[1] 0.9295474
> meanrecall=mean(recall)
> meanrecall
[1] 0.9864615
```

KNN on 226 entries with 100 replications:

KNN is a non-parametric method used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space.

```
for (k in 1:rep) {
+     model = sample(1:TotalEntries,Training)
+     model.Train = datasubset[model,2:8]
+     model.Test = datasubset[-model,2:8]
+     model.trainLabels <- datasubset[model,9]
```

```
+     datasubset.knn3 = knn(model.Train,model.Test,model.trainLabels,k=3)
+     tablin=table(datasubset.knn3,datasubset[-model,9])
+     errlin[k] = (TotalEntries - sum(diag(tablin)))/TotalEntries
+     accuracy[k]=sum(diag(tablin))/TotalEntries
+     precision[k]=tablin[1,1]/sum(tablin[1,1:2])
+     recall[k]=tablin[1,1]/sum(tablin[1:2,1])
+ }
> meanaccuracy=mean(accuracy)
> meanaccuracy
```

[1] 0.945

```
> meanprecision=mean(precision)
> meanprecision
```

[1] 0.9598977

```
> meanrecall=mean(recall)
> meanrecall
```

[1] 0.9641075

Summary:

Model	Accuracy	Precision	Recall
Logistic Regression	0.9514	0.9670328	0.9712895
LDA	0.9312	0.9474137	0.9645754
QDA	0.9354	0.9295474	0.9864615
KNN(k=3)	0.945	0.9598977	0.9641075

Analysis Summary:

1. As from the above values when taking Accuracy in to the consideration for the KC Weather data set the **Logistic Regression** has high accuracy compared to other classification models. If the business requires to build a model by taking only accuracy in to the consideration, then the Logistic Regression model is the best fit.
2. QDA allows for different variances among the classes, as the resulting accuracy and recall has high values compared to LDA. So QDA performs better than LDA for the KC weather data.
3. By the above values, KNN can be a better model than the LDA as the accuracy level is compared.

1.a. ii. Discuss and analyze in a systematic way you would consider eliminating some of the predictors and see if your accuracy, precision and recall improves

Considering all the predictors:

```
> model<-glm(Events~Temp.F+Dew_Point.F+Humidity.percentage+Sea_Level_Press.in+Visibility.mi+Wind.mph+Precip
.in, training,family="binomial")
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(model)
Call:
glm(formula = Events ~ Temp.F + Dew_Point.F + Humidity.percentage +
    Sea_Level_Press.in + Visibility.mi + Wind.mph + Precip.in,
    family = "binomial", data = training)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.74830  -0.00506  -0.00002   0.00000   1.49922

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)         -262.3491   156.2865  -1.679   0.0932 .
Temp.F                 0.3192     0.3550   0.899   0.3685
Dew_Point.F           -0.8271     0.4648  -1.780   0.0752 .
Humidity.percentage    0.2277     0.1877   1.213   0.2250
Sea_Level_Press.in     8.5154     5.2930   1.609   0.1077
Visibility.mi         -0.4256     0.5737  -0.742   0.4581
Wind.mph               0.3266     0.2042   1.600   0.1096
Precip.in           -180.4326    91.6852  -1.968   0.0491 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 181.019  on 175  degrees of freedom
Residual deviance:  24.515  on 168  degrees of freedom
AIC: 40.515

Number of Fisher Scoring iterations: 12
```

```
> res<-predict(model,testing,type="response")
> (table(ActualValue=testing$Events, PredictedValue=res>0.5))
                    PredictedValue
ActualValue         FALSE TRUE
  Rain                 36    1
  Snow                  0   13
```

Removing Visibility:

```
> model<-glm(Events~Temp.F+Dew_Point.F+Humidity.percentage+Sea_Level_Press.in+Wind.mph+Precip.in, training,family="bino
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(model)

Call:
glm(formula = Events ~ Temp.F + Dew_Point.F + Humidity.percentage +
    Sea_Level_Press.in + Wind.mph + Precip.in, family = "binomial",
    data = training)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.81890  -0.00659  -0.00002   0.00000   1.36365

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)         -232.8111   143.9595  -1.617   0.1058
Temp.F                 0.4378     0.3319   1.319   0.1871
Dew_Point.F           -0.9676     0.4431  -2.184   0.0290 *
Humidity.percentage    0.3065     0.1630   1.881   0.0600 .
Sea_Level_Press.in     7.2308     4.6979   1.539   0.1238
Wind.mph               0.2580     0.1776   1.453   0.1463
Precip.in           -165.0166    83.9478  -1.966   0.0493 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 181.019  on 175  degrees of freedom
Residual deviance:  25.114  on 169  degrees of freedom
AIC: 39.114

Number of Fisher Scoring iterations: 12
> res<-predict(model,testing,type="response")
> (table(ActualValue=testing$Events, PredictedValue=res>0.5))
                  PredictedValue
ActualValue         FALSE TRUE
  Rain                 36    1
  Snow                  0   13
```

Removing Temp.F:

```
> model<-glm(Events~Dew_Point.F+Humidity.percentage+Sea_Level_Press.in+Visibility.mi+Wind.mph+Precip.in, training,famil
omial")
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(model)

Call:
glm(formula = Events ~ Dew_Point.F + Humidity.percentage + Sea_Level_Press.in +
    Visibility.mi + Wind.mph + Precip.in, family = "binomial",
    data = training)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.60942  -0.00581  -0.00004   0.00000   1.55288

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)         -274.10090  147.29736  -1.861  0.06276 .
Dew_Point.F           -0.44950    0.14173  -3.172  0.00152 **
Humidity.percentage    0.07113    0.06991   1.017  0.30895
Sea_Level_Press.in     9.37899    5.01403   1.871  0.06141 .
Visibility.mi         -0.66798    0.53689  -1.244  0.21344
Wind.mph               0.37485    0.19494   1.923  0.05450 .
Precip.in           -172.98412   91.14584  -1.898  0.05771 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 181.019  on 175  degrees of freedom
Residual deviance:  25.416  on 169  degrees of freedom
AIC: 39.416

Number of Fisher Scoring iterations: 12

> res<-predict(model,testing,type="response")
> (table(ActualValue=testing$Events, PredictedValue=res>0.5))
                  PredictedValue
ActualValue         FALSE TRUE
```

```
  Rain                    36    1
  Snow                     0   13
```

```
Removing Humidity:

> model<-glm(Events~Temp.F+Dew_Point.F+Sea_Level_Press.in+Visibility.mi+Wind.mph+Precip.in, training,family
="binomial")
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(model)

Call:
glm(formula = Events ~ Temp.F + Dew_Point.F + Sea_Level_Press.in +
    Visibility.mi + Wind.mph + Precip.in, family = "binomial",
    data = training)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.63405  -0.00690  -0.00006   0.00000   1.57666

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)        -306.77954  146.64036  -2.092   0.0364 *
Temp.F               -0.07343    0.12254  -0.599   0.5490
Dew_Point.F          -0.34159    0.16043  -2.129   0.0332 *
Sea_Level_Press.in   10.67704    4.94185   2.161   0.0307 *
Visibility.mi        -0.87181    0.50457  -1.728   0.0840 .
Wind.mph              0.37703    0.19485   1.935   0.0530 .
Precip.in          -164.40139   94.80374  -1.734   0.0829 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 181.019  on 175  degrees of freedom
Residual deviance:  26.164  on 169  degrees of freedom
AIC: 40.164

Number of Fisher Scoring iterations: 12

> res<-predict(model,testing,type="response")
> (table(ActualValue=testing$Events, PredictedValue=res>0.5))
                   PredictedValue
ActualValue          FALSE TRUE
  Rain                  36    1
  Snow                   0   13

Summary:
```

	Residual deviance	AIC	Accuracy	Precision	Recall
All predictors	24.515	40.515	0.9801	1.00	0.97297
All Predictors-Visibility	25.114	39.114	0.9801	1.00	0.97297
All Predictors-Temperature	25.416	39.416	0.9801	1.00	0.97297
All Predictors-Humidity	26.164	40.164	0.9801	1.00	0.97297

Analysis summary:

In the case of excluding the predictors Visibility, Temperature and Humidity all the cases AIC values are lesser when compared considering all the predictors.

Hence, **Visibility, Temperature and Humidity are not significant predictors** for the model as the **AIC value is decreasing** when compared to model where all predictors are included. Thus the accuracy and precision can be improvise.

1.b. Consider next the entire dataset consisting of 366 entries. Now logistics regression cannot be applied, but you can apply the rest of them. Repeat the above studies in i) and ii) with LDA, QDA, and knn on the entire data set (using 290 of them in a training set). Do not forget randomization and 100 replications for this study.

1.b.i Solution:

LDA on 366 entries with 100 replications:

```
> TotalEntries=366
> Training=290
> Testing=TotalEntries-Training
> rep=100
> accuracy=dim(rep)
>
> precision_rain=dim(rep)
> precision_rain_thunderstrom=dim(rep)
> precision_snow=dim(rep)
>
> recall_rain =dim(rep)
> recall_rain_thunderstrom=dim(rep)
```

```
> recall_snow=dim(rep)
>
> for(k in 1:rep){
+     train=sample(1:TotalEntries,Training)
+     data.lda_train=lda(Events~Temp.F+Dew_Point.F+Humidity.percentage+Sea_Level_Press.in+Visibility.mi+Win
d.mph+Precip.in,data[train,])
+     predict(data.lda_train,data[-train,])$class
+     tablin=table(data$Events[-train],predict(data.lda_train,data[-train,])$class)
+     errlin[k]=(Testing-sum(diag(tablin)))/Testing
+     accuracy[k]=sum(diag(tablin))/Testing
+     precision_rain[k] = (tablin[1,1])/(tablin[1,1]+tablin[2,1]+tablin[3,1])
+     precision_rain_thunderstrom[k] = (tablin[2,2])/(tablin[1,2]+tablin[2,2]+tablin[3,2])
+     precision_snow[k] = (tablin[3,3])/(tablin[1,3]+tablin[2,3]+tablin[3,3])
+     recall_rain[k]=(tablin[1,1])/(tablin[1,1]+tablin[1,2]+tablin[1,3])
+     recall_rain_thunderstrom[k]=(tablin[2,2])/(tablin[2,1]+tablin[2,2]+tablin[2,3])
+     recall_snow[k]=(tablin[3,3])/(tablin[3,1]+tablin[3,2]+tablin[3,3])
+ }
> print(mean(errlin))
[1] 0.2519737
> print(mean(accuracy))
[1] 0.7480263
> print(mean(precision_rain))
[1] 0.7523023
> print(mean(precision_rain_thunderstrom))
[1] 0.7237916
> print(mean(precision_snow))
[1] 0.8021281
> print(mean(recall_rain))
[1] 0.7087325
> print(mean(recall_rain_thunderstrom))
[1] 0.7499575
> print(mean(recall_snow))
[1] 0.8945227
```

QDA on 366 entries with 100 replications:

```
> for(k in 1:rep){
+     train=sample(1:TotalEntries,Training)
+     data.lda_train=qda(Events~Temp.F+Dew_Point.F+Humidity.percentage+Sea_Level_Press.in+Visibility.mi+Win
d.mph+Precip.in,data[train,])
+     predict(data.lda_train,data[-train,])$class
+     tablin=table(data$Events[-train],predict(data.lda_train,data[-train,])$class)
+     errlin[k]=(Testing-sum(diag(tablin)))/Testing
+     accuracy[k]=sum(diag(tablin))/Testing
+     precision_rain[k] = (tablin[1,1])/(tablin[1,1]+tablin[2,1]+tablin[3,1])
+     precision_rain_thunderstrom[k] = (tablin[2,2])/(tablin[1,2]+tablin[2,2]+tablin[3,2])
+     precision_snow[k] = (tablin[3,3])/(tablin[1,3]+tablin[2,3]+tablin[3,3])
+     recall_rain[k]=(tablin[1,1])/(tablin[1,1]+tablin[1,2]+tablin[1,3])
+     recall_rain_thunderstrom[k]=(tablin[2,2])/(tablin[2,1]+tablin[2,2]+tablin[2,3])
+     recall_snow[k]=(tablin[3,3])/(tablin[3,1]+tablin[3,2]+tablin[3,3])
+ }
> print(mean(errlin))
[1] 0.255
> print(mean(accuracy))
[1] 0.745
> print(mean(precision_rain))
[1] 0.7498849
> print(mean(precision_rain_thunderstrom))
[1] 0.7184631
> print(mean(precision_snow))
[1] 0.7964458
> print(mean(recall_rain))
[1] 0.7016425
> print(mean(recall_rain_thunderstrom))
[1] 0.7264472
> print(mean(recall_snow))
[1] 0.9515166
```

KNN on 366 entries with 100 replications:

```
    > for (k in 1:rep) {
+     model = sample(1:TotalEntries,Training)
+     model.Train = data[model,2:8]
+     model.Test = data[-model,2:8]
+     model.trainLabels <- data[model,9]
+     data.knn3 = knn(model.Train,model.Test,model.trainLabels,k=3)
+     tablin=table(data.knn3,data[-model,9])
+     errlin[k] = (TotalEntries - sum(diag(tablin)))/TotalEntries
+     accuracy[k]=sum(diag(tablin))/Testing
+     precision[k]=tablin[1,1]/sum(tablin[1,1:3])
+     precision_rain[k] = (tablin[1,1])/(tablin[1,1]+tablin[2,1]+tablin[3,1])
+     precision_rain_thunderstrom[k] = (tablin[2,2])/(tablin[1,2]+tablin[2,2]+tablin[3,2])
+     precision_snow[k] = (tablin[3,3])/(tablin[1,3]+tablin[2,3]+tablin[3,3])
+     recall_rain[k]=(tablin[1,1])/(tablin[1,1]+tablin[1,2]+tablin[1,3])
+     recall_rain_thunderstrom[k]=(tablin[2,2])/(tablin[2,1]+tablin[2,2]+tablin[2,3])
+     recall_snow[k]=(tablin[3,3])/(tablin[3,1]+tablin[3,2]+tablin[3,3])
+ }
> print(mean(errlin))
[1] 0.847623
> print(mean(accuracy))
[1] 0.7338158
> print(mean(precision_rain))
[1] 0.7262863
> print(mean(precision_rain_thunderstrom))
[1] 0.6946528
> print(mean(precision_snow))
[1] 0.8897852
> print(mean(recall_rain))
[1] 0.7250768
> print(mean(recall_rain_thunderstrom))
[1] 0.6982034
```

```
> print(mean(recall_snow))
[1] 0.8713335
```

Summary:

For multi-class classification precision and recall are calculated as below for all Predictors

Model	Error	Accuracy	Precision			Recall		
			Rain	Rain_Thunderstorm	Snow	Rain	Rain_Thunderstorm	Snow
LDA	0.2519737	0.7480263	0.7523023	0.7237916	0.8021281	0.7087325	0.7499575	0.8945227
QDA	0.255	0.745	0.7498849	0.7184631	0.7964458	0.7016425	0.7264472	0.9515166
KNN(k=3)	0.847623	0.733815	0.7262863	0.6946528	0.8897852	0.7250768	0.6982034	0.8713335

Analysis Summary:

1. As from the above values taking in to the consideration the parameters Error rate and accuracy. Higher the rate of accuracy and less error rate then the model is a better fit. LDA outperforms when compared to other models.
2. For multi classification the precision and recall values are calculated for each class. In the KC Weather data set there are particularly three classes Rain, Rain Thunder Strom, Snow.
3. When the business need to implement the better model fit depending on the Accuracy consideration then the LDA out performs compared to QDA and KNN.

1.b.ii Analyze in a systematic way you would consider eliminating some of the predictors and see if your accuracy, precision and recall improves.

By removing the predictor **Temp** from the given data set and performing LDA, QDA,KNN

LDA:

```
for(k in 1:rep){
+     train=sample(1:TotalEntries,Training)
+     data.lda_train=lda(Events~Temp.F+Dew_Point.F+Humidity.percentage+Sea_Level_Press.in+Wind.mph+Precip.i
n,data[train,])
+     predict(data.lda_train,data[-train,])$class
+     tablin=table(data$Events[-train],predict(data.lda_train,data[-train,])$class)
+     errlin[k]=(Testing-sum(diag(tablin)))/Testing
+     accuracy[k]=sum(diag(tablin))/Testing
+     precision_rain[k] = (tablin[1,1])/(tablin[1,1]+tablin[2,1]+tablin[3,1])
+     precision_rain_thunderstrom[k] = (tablin[2,2])/(tablin[1,2]+tablin[2,2]+tablin[3,2])
+     precision_snow[k] = (tablin[3,3])/(tablin[1,3]+tablin[2,3]+tablin[3,3])
+     recall_rain[k]=(tablin[1,1])/(tablin[1,1]+tablin[1,2]+tablin[1,3])
+     recall_rain_thunderstrom[k]=(tablin[2,2])/(tablin[2,1]+tablin[2,2]+tablin[2,3])
+     recall_snow[k]=(tablin[3,3])/(tablin[3,1]+tablin[3,2]+tablin[3,3])
+ }
> print(mean(errlin))
[1] 0.2511842
> print(mean(accuracy))
[1] 0.7488158
> print(mean(precision_rain))
[1] 0.7542738
> print(mean(precision_rain_thunderstrom))
[1] 0.7158733
> print(mean(precision_snow))
[1] 0.8320023
> print(mean(recall_rain))
[1] 0.7081035
> print(mean(recall_rain_thunderstrom))
[1] 0.7444648
> print(mean(recall_snow))
[1] 0.9227971
```

QDA:

```
for(k in 1:rep){
+     train=sample(1:TotalEntries,Training)
+     data.lda_train=qda(Events~Temp.F+Dew_Point.F+Humidity.percentage+Sea_Level_Press.in+Wind.mph+Precip.i
n,data[train,])
+     predict(data.lda_train,data[-train,])$class
+     tablin=table(data$Events[-train],predict(data.lda_train,data[-train,])$class)
+     errlin[k]=(Testing-sum(diag(tablin)))/Testing
+     accuracy[k]=sum(diag(tablin))/Testing
+     precision_rain[k] = (tablin[1,1])/(tablin[1,1]+tablin[2,1]+tablin[3,1])
+     precision_rain_thunderstrom[k] = (tablin[2,2])/(tablin[1,2]+tablin[2,2]+tablin[3,2])
+     precision_snow[k] = (tablin[3,3])/(tablin[1,3]+tablin[2,3]+tablin[3,3])
+     recall_rain[k]=(tablin[1,1])/(tablin[1,1]+tablin[1,2]+tablin[1,3])
+     recall_rain_thunderstrom[k]=(tablin[2,2])/(tablin[2,1]+tablin[2,2]+tablin[2,3])
+     recall_snow[k]=(tablin[3,3])/(tablin[3,1]+tablin[3,2]+tablin[3,3])
+ }
> print(mean(errlin))
[1] 0.2502632
> print(mean(accuracy))
[1] 0.7497368
> print(mean(precision_rain))
```

```
[1] 0.7564906
> print(mean(precision_rain_thunderstrom))
[1] 0.7307324
> print(mean(precision_snow))
[1] 0.7785842
> print(mean(recall_rain))
[1] 0.7045744
> print(mean(recall_rain_thunderstrom))
[1] 0.7407665
> print(mean(recall_snow))
[1] 0.9413258
```

KNN:

```
> for (k in 1:rep) {
+     model = sample(1:TotalEntries,Training)
+     model.Train = data[model,3:8]
+     model.Test = data[-model,3:8]
+     model.trainLabels <- data[model,9]
+     data.knn3 = knn(model.Train,model.Test,model.trainLabels,k=3)
+     tablin=table(data.knn3,data[-model,9])
+     errlin[k] = (TotalEntries - sum(diag(tablin)))/TotalEntries
+     accuracy[k]=sum(diag(tablin))/Testing
+     precision[k]=tablin[1,1]/sum(tablin[1,1:3])
+     precision_rain[k] = (tablin[1,1])/(tablin[1,1]+tablin[2,1]+tablin[3,1])
+     precision_rain_thunderstrom[k] = (tablin[2,2])/(tablin[1,2]+tablin[2,2]+tablin[3,2])
+     precision_snow[k] = (tablin[3,3])/(tablin[1,3]+tablin[2,3]+tablin[3,3])
+     recall_rain[k]=(tablin[1,1])/(tablin[1,1]+tablin[1,2]+tablin[1,3])
+     recall_rain_thunderstrom[k]=(tablin[2,2])/(tablin[2,1]+tablin[2,2]+tablin[2,3])
+     recall_snow[k]=(tablin[3,3])/(tablin[3,1]+tablin[3,2]+tablin[3,3])
+ }
>
> print(mean(errlin))
[1] 0.8433333
> print(mean(accuracy))
[1] 0.7544737
> print(mean(precision_rain))
[1] 0.728611
> print(mean(precision_rain_thunderstrom))
[1] 0.7386631
> print(mean(precision_snow))
[1] 0.898578
> print(mean(recall_rain))
[1] 0.7489206
> print(mean(recall_rain_thunderstrom))
[1] 0.7123012
> print(mean(recall_snow))
[1] 0.9048301
```

Summary:

Outputs when the Predictor Temp is removed.

Model	Error	Accuracy	Precision			Recall		
			Rain	Rain_Thunderstorm	Snow	Rain	Rain_Thunderstorm	Snow
LDA	0. 2511842	0. 7488158	0. 7542738	0. 7158733	0. 8320023	0. 7081035	0. 7444648	0. 9227971
QDA	0. 2502632	0. 7497368	0. 7564906	0. 7307324	0. 7785842	0. 7045744	0. 7407665	0. 9413258
KNN(k=3)	0. 8433333	0. 7544737	0. 728611	0. 7386631	0. 898578	0. 7489206	0. 7123012	0. 9048301

Analysis Summary:

With all predictors

Model	Error	Accuracy	Precision			Recall		
			Rain	Rain_Thunderstorm	Snow	Rain	Rain_Thunderstorm	Snow
LDA	0.2519737	0.7480263	0.7523023	0.7237916	0.8021281	0.7087325	0.7499575	0.8945227
QDA	0.255	0.745	0.7498849	0.7184631	0.7964458	0.7016425	0.7264472	0.9515166
KNN(k=3)	0.847623	0.733815	0.7262863	0.6946528	0.8897852	0.7250768	0.6982034	0.8713335

With comparing the actual predictors to the outputs when the predictor is removed -Can be conclude that the Error rate and accuracy improved. The error rate is reduced when the predictor is removed ,and the accuracy has improved .Thus the temp has less significance and model without temp outperforms well than the model with all predictors in LDA,QDA,KNN.