USC
EE559 Final Project

Algerian forest fires

Dheeraj Panneer Selvam, dpanneer@usc.edu

2 May 2022

1. Abstract

My project's goal was to predict if there will be a fire in a future date given the weather conditions. My dataset contained of 10 features including the date and the output label which indicated if there will be a fire or not. I tried to tackle this problem by creating and comparing 7 models including the trivial and baseline models namely trivial, N-means, Bayes Minimum error classifier, Perceptron algorithm with Sequential GD and Scheduler, MSE LMS algorithm with sequential GD and Scheduler, SVM Linear Classifier, SVM nonlinear RBF kernel classifier. In addition, I performed certain preprocessing, feature expansion, feature selection on the dataset to maximize the accuracies I got on the models.

The best accuracy achieved was 95% by MSE but that was unreliable as the accuracies kept varying due to randomly shuffling the dataset. Second highest I got was 91.6% on the perceptron model, but again it varied a lot due to the randomness of shuffling. The best reliable model was the Bayes model which gave me an accuracy of 88.33% and F1 score of 0.844 which I would choose among the 3 best models I was able to create.

2. Introduction

2.1. Problem Statement and Goals

My Dataset is the Algerian Forest Fires, which is a time series weather dataset giving info about the weather conditions over a period of 3 months. It contains dates, 8 features namely Temperature, Humidity, Wind Speed, Rain, FFMC, DMC, DC, ISI, BUI and the output labels which tells us if there was a fire or not. The goal is to predict if there will be a fire in the future dates based on the weather conditions.

2.2. Literature Review

Most of the work was based on the material and homework of EE559. No work was referred, some blogs about time series data was looked up. (https://medium.com/swlh/time-series-analysis-7006ea1c3326)

3. Approach and Implementation

3.1. Dataset Usage

3.1.1. Splitting Default Train Set

The train set was split into 75% train and 25% validation, that is 184 datapoints were split into 138 datapoints in train and 46 datapoints in validation, so in total I divided the dataset into 4 parts of 46 datapoints each and performed cross validation. For cross validation I used one for loop, within it I first assigned the validation set for each epoch and dropped the validation set from the original set to create a train set, then check for the accuracy on the validation set across the 4 epochs.

3.1.2. Default Test Set Usage

After Cross-validation, all models best validation set accuracy was around 92%, I checked the test set accuracy for the train set with the best validation set accuracy, no modifications were done on test set. Later I checked how much change occurs when I train the model on the entire train set without splitting into train/validation and used test set for checking the accuracy, again no modifications were made on test or train.

3.1.2. Expanding Dataset

The train set and test set was later expanded with other manually created features explained in the below subsections. But due to features containing data from the past 7 days were created by me, I had to remove the last 7 days entries on the train set as the test set will contain info about them and affect the training. So, the train set was reduced to 167 datapoints from 184. I had taken data pertaining to the previous 5 days and 2 days as well, but as I am removing the last 7 days entry, these will automatically be removed.

3.1.3. Cross-Validation Expanded feature space

As mentioned above we must make sure data is not leaked into other sets, so while performing cross validation on expanded feature space, I used same technique as mentioned above but tweaked it a bit. I assigned the validation set first but made sure to drop off last 7 days entry in it. The I dropped all the validation entries from the main dataset and 7 days entry before the start date of the validation set. By doing so the train set and validation set does not have info on the other. Again, CV was performed on all the models to get the validation set accuracy. The number of datapoints keeps varying in train and validation as we are removing the last 7 days entry. Usually, validation set has around 32 datapoints and train has around 105 datapoints.

3.1.4. Expanded Test Set Usage

I used the expanded test set to check for accuracy of models based on train sets with highest validation set accuracy. Later I trained the entire expanded tarin set no train/valid split and checked their accuracy.

3.1.5. Selected Features

The expanded feature space was reduced to only the selected features based on techniques mentioned in other subsections. The train set and test set were reduced to the selected number of features and the entire new train set was used to check accuracy on the test set. Cross-Validation was not used, I performed analysis on all models after selecting the features.

3.2. Preprocessing

I type casted the dates from str format to datetime format, this was done to make my work easy while choosing data pertaining to certain range of dates, that was majorly done during feature expansion, selection and cross-validation.

Besides that, at the end of finalizing the model, I used Min-Max normalization to normalize my feature selected dataset, to check if there was improvement in the performance, it had a negative effect on the selected models, so I excluded the normalization process.

I even tried standardizing the feature selected data, but the effect was negative, so I excluded standardization as well.

The results of normalizing and standardizing are provided in the results and comparison section.

3.3. Feature engineering and Feature dimensionality adjustment

3.3.1. Feature Expansion

I used the major weather features namely Temperature, Rain, Humidity and Wind Speed took their mean, median, max and min values over the past 7 days, 5 days and 2 days. In total including the date and output column there were 10+(3*4*4) = 58 features, that is 48 additional features.

According to the thumb rule N > (3 – 10) D+1, which means for N = 184, the features should be in the range of 17 – 61 features. After expansion I have 58 features which falls exactly in this range. So, I can train the models on the entire expanded dataset as well and I tried this the results are shown in other section.

3.3.2. Feature Selection

Even if my expanded feature space was in the correct range of dimensions, the accuracy and f1 score obtained was not satisfactory for me so I decided to do feature selection based on two methods.

Pearson's co-relation coefficients:

I used pandas to get the correlation matrix of all the 57 features excluding date, then I iterated through all the 56 features correlation values with respect to classes the output label. After that I extracted all the features having a minimum threshold value. I experimented with various values for various models, the best minimum threshold value was 0.33, for most models, especially for the Perceptron and MSE models. At 0.33 the model had 16 features which is almost in the desired range. Other models performed above average to average with this method of feature selection. So, I tried a different method to see if I get better accuracies for the various models.

Individual Features accuracy-based selection:

I iterated through all the 56 individual features and found the accuracy of each model based on that one single feature and stored all the features which had an above threshold accuracy which was given by me. For most models I gave a threshold accuracy of 65%, the bayes model and perceptron model performed well in this feature selection yielding close to 90% accuracy. The number of features varied from model to model from 6 – 25. But as the accuracy and f1 scores were good this was overruled.

Improvements:

I was about to perform sequential feature selection, that is modify the above function to iterate through all features and keep adding a feature until highest accuracy is achieved, which is like 56*56 iterations. Due to time constrains and as I had already achieved accuracies around 90%, I dropped this method.

3.4. Training, classification and/or regression, and model selection

3.4.1. Trivial Model and Baseline Model

The Trivial model was build using the overall probability of if the fire will occur or not, it has nothing to do with the datapoints and randomly generates labels based on the probabilities. This model was used to check if our models even learned anything. Accuracy on test ranged from 45-60%.
The Baseline model was the N-means classifier, which took the mean of datapoints belonging to a particular class and the Euclidian distance was calculated from each of the means to the new datapoint's and the label was assigned based on class with the closest mean.

3.4.2. Bayes Minimum Error Classifier (Gaussian Distribution Model)

The bayes model was based on the bayes theorem and gaussian distribution model. The decision boundary of the gaussian distribution model is calculated using the covariance matrix, the means and probability of occurrence of the class. As this is a statistical based models there are no weight vectors and decisions are made based on the decision boundaries. First, I calculated the discriminant function of each class based on the equation
gi = -(1/2)ln(|Ei|)-(1/2)(X-mi)TE/^-1(X-mi)+ln(P(Si)). Later used maximum value method to classify the labels. As this model is not based on weights there are no hyperparameters for tunning.

3.4.3. Perceptron Learning Algorithm (Sequential GD and Scheduler for Eta)

The perceptron model was based on sequential gradient descent and a scheduler to choose the optimal eta. The model was split across 4 functions one was used to calculate the cost J(w) based on the rule. Another function which calculated the optimal weight based on eta and sequential gradient decent. The initial weights were randomly generated from -0.05 to 0.05, this range was provided based on trial and error after certain tries. The dataset was randomly shuffled once, instead of each iteration to avoid fluctuations in accuracies. Eta was given by a/(b+i) for each iteration, a = [0.01,0.1,1,10,100] and b = [1,10,100,1000], overall, 20 combinations of eta were given for each i, the 3rd function was used to get the labels based on the optimal weights and finally the main function which made use of all helper functions. There was no external parameter control for this model, and it ran through the entire process when called and each time found the best eta for the dataset.

3.4.4. Mean Square Error Classifier (Widrow-Hoff) based on Sequential GD and Scheduler

The MSE model was based on the LMS or Widrow-Hoff algorithm in combination with sequential gradient decent and a scheduler to find optimal eta. Again, split across 4 functions, one to calculated J(w) based on the rule. Second to calculated optimal weights. The initial weights were randomly generated from -0.2 to 0.2, this range was provided based on trial and error after certain tries. The dataset was randomly shuffled once, instead of each iteration to avoid fluctuations in accuracies. Eta was again given by a/(b+i), this time a was set to a constant = 0.01 and b was varied for [0.01,0.1,1,10,100,1000]. Overall, 6 combinations for each i. The 3rd function used the optimal weights obtained to get the class labels. The main function used all the helper function to define the model. There was no external parameter control for this model, and it ran through the entire process when called and each time found the best eta for the dataset.

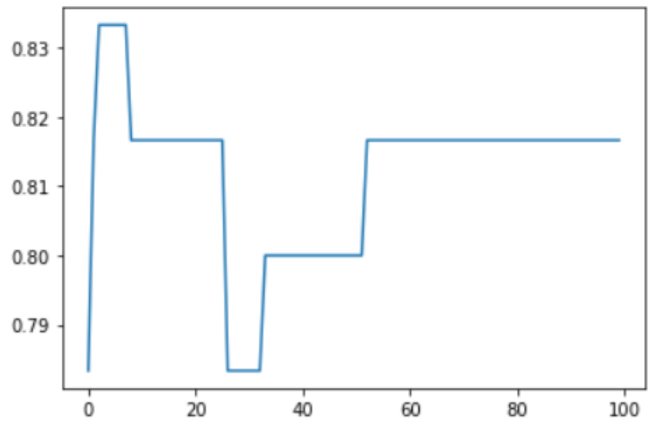
3.4.5. SVM Linear Classifier

This the basic Linear SVM classifier. I converted the equations into the matrix form of Al = p, where A is the derived matrix, l is lambada and p is the output value. I calculated A, by first assigning a n*n matrix and then filled it up using nested loops based on the derived equation that is

$A_{ij} = z_i * z_j * u_i^T * u_j$, then added all -z values to n+1 column and z values to n+1 row, thus forming the A matrix. p was a vector of ones and last element was 0. I calculated lambada using A and p. Then I obtained the weights $w_n = z_n * \ln * X_n$ and obtained the bias weight w0 = $(1/z_i)-(w^T*X_i)$. Using the weights, I found the decision boundary and classified the labels. As this was the linear model there were no parameters to control.

3.4.6. SVM Nonlinear Classifier (RBF Kernel)

The method used for this model was same as the SVM linear model, but while calculating the A matrix, I introduced RBF kernel instead of the linear model i.e $A_{ij} = z_i * z_j * k(x',x)$. Where k(x',x) is the RBF kernel that is $\exp\{-gamma*||x'-x||_2^2\}$. The reaming process was the same. For the RBF I calculated the optimal gamma value based on the test set directly. I choose gamma values from 0.001 up to 0.2 in steps of 0.02 totally trying 100 gamma values. The optimal gamma value was around 0.01.



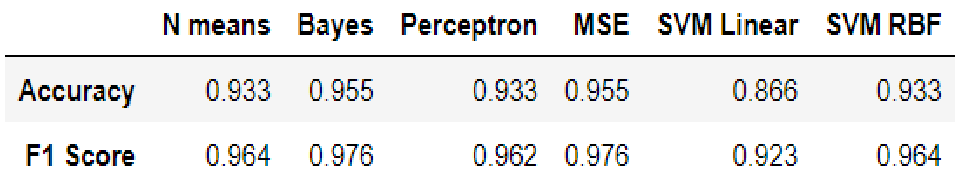
4. Results and Analysis: Comparison and Interpretation

For all sections below the **trivial system** gave an **accuracy between 45-60%** and an **F1 score** in the same range of **0.45-0.6**. So, I have not included trivial regressor outputs in below tables as the outputs were random and were just to check if other models are working.

*Note – The confusion matrix for all the outputs shown can be seen in the code pdf. I have included only the major ones that is final trivial, final n-means and final best model in the report as others are unnecessary. *

4.1. Cross-Validation on Base Dataset

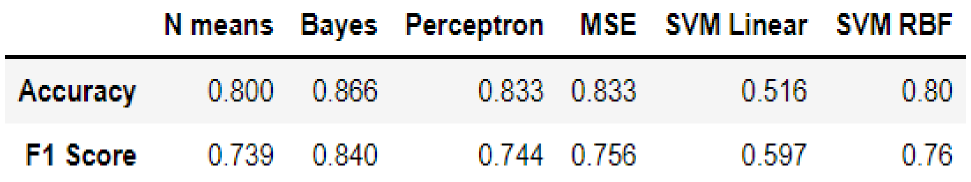
4.1.1. Validation Sets Best Accuracy and F1 score



	N means	Bayes	Perceptron	MSE	SVM Linear	SVM RBF
Accuracy	0.933	0.955	0.933	0.955	0.866	0.933
F1 Score	0.964	0.976	0.962	0.976	0.923	0.964

As seen in the table the best accuracies and f1 scores among the validation sets or epochs is given all model are at least at 93% accuracy, except SVM linear. So, I decided to use the test set for accuracy.

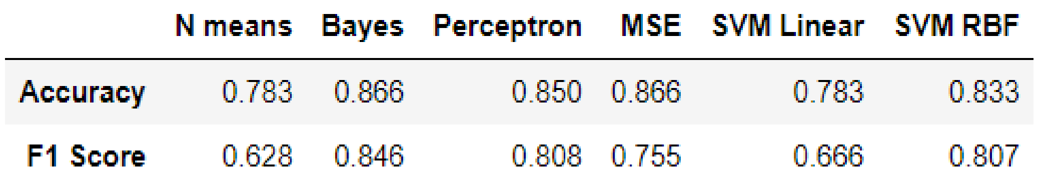
4.1.2. Test set accuracy and F1 score on split train set (75% of original train)



	N means	Bayes	Perceptron	MSE	SVM Linear	SVM RBF
Accuracy	0.800	0.866	0.833	0.833	0.516	0.80
F1 Score	0.739	0.840	0.744	0.756	0.597	0.76

As most of the validation accuracies were same, I used the test set to check the accuracy and f1 scores based on the partial split train set that is 75 % of the original given train set of the best epoch i.e., the train set which gave the highest accuracy on validation set. Bayes model gave the best result followed by MSE and Perceptron. It is to be noted that for MSE and Perceptron models the accuracies varies as I was using sequential GD which randomly shuffles the data for each iteration.

4.2. Test set Accuracy and F1 score on entire train set (no split 100% train)

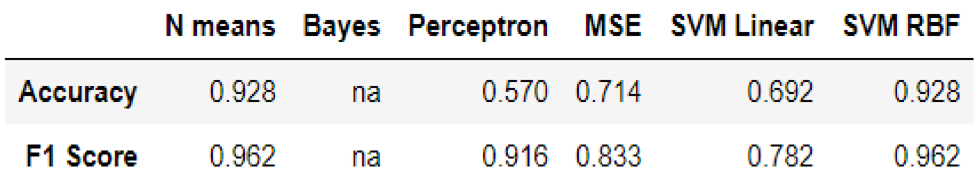


	N means	Bayes	Perceptron	MSE	SVM Linear	SVM RBF
Accuracy	0.783	0.866	0.850	0.866	0.783	0.833
F1 Score	0.628	0.846	0.808	0.755	0.666	0.807

I was curious if the model's accuracies would improve if I had trained them over the entire dataset and thus tried doing so. The bayes model remained the same were as the perceptron and MSE models saw an increase in performance. The SVM models were below par compared to other models. But I kept them for reference and see if any improvements occur in the following process.

4.3. Cross-Validation on expanded feature space

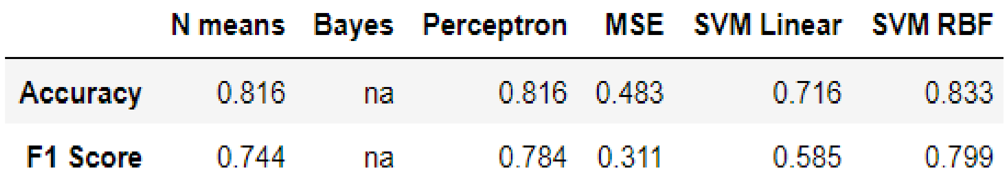
4.3.1. Expanded Validation Sets Best Accuracy and F1 score



	N means	Bayes	Perceptron	MSE	SVM Linear	SVM RBF
Accuracy	0.928	na	0.570	0.714	0.692	0.928
F1 Score	0.962	na	0.916	0.833	0.782	0.962

I was not able to perform cross validation on Bayes model because when I took the determinant of E it became too small for certain epochs and caused a math domain error while using log on det(E), so I skipped cv for the bayes model. Surprisingly, the best models were SVM RBF and N means. But as this was the entire feature expanded model, I just wanted to test the models before feature selection to see how they perform.

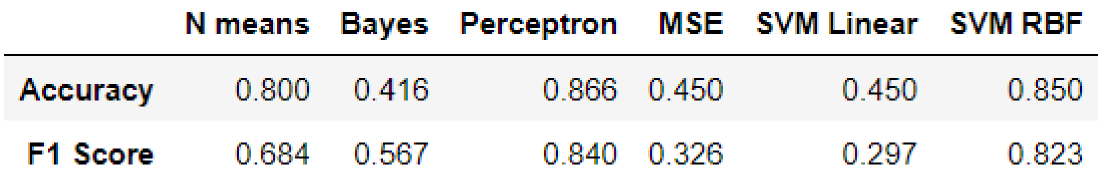
4.3.2. Expanded Test set accuracy and F1 score on split train set (75% of original train)



	N means	Bayes	Perceptron	MSE	SVM Linear	SVM RBF
Accuracy	0.816	na	0.816	0.483	0.716	0.833
F1 Score	0.744	na	0.784	0.311	0.585	0.799

As done previously I wanted to check the test set accuracies for these models for the 75% train set of the best epoch and compare them with the previous results on base dataset just for interpretation purposes and check how all models work. No results were satisfactory when I used the entire expanded feature space for training.

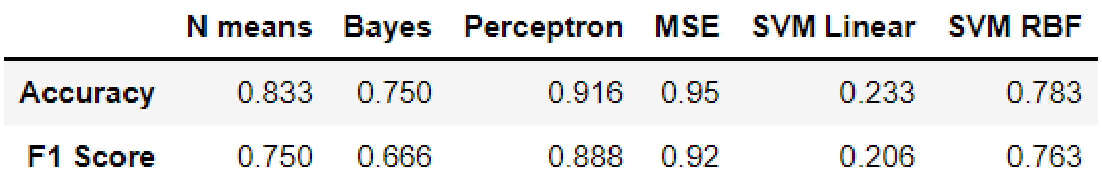
4.4. Expanded Test set Accuracy and F1 score on entire train set (no split 100% train)



	N means	Bayes	Perceptron	MSE	SVM Linear	SVM RBF
Accuracy	0.800	0.416	0.866	0.450	0.450	0.850
F1 Score	0.684	0.567	0.840	0.326	0.297	0.823

Checking the test set error for the models trained over the entire 100% expanded training set, I noticed that Bayes model was counterproductive and other models were performing almost at the same level as before. So, I proceeded to do feature selection and optimize the models for better accuracies.

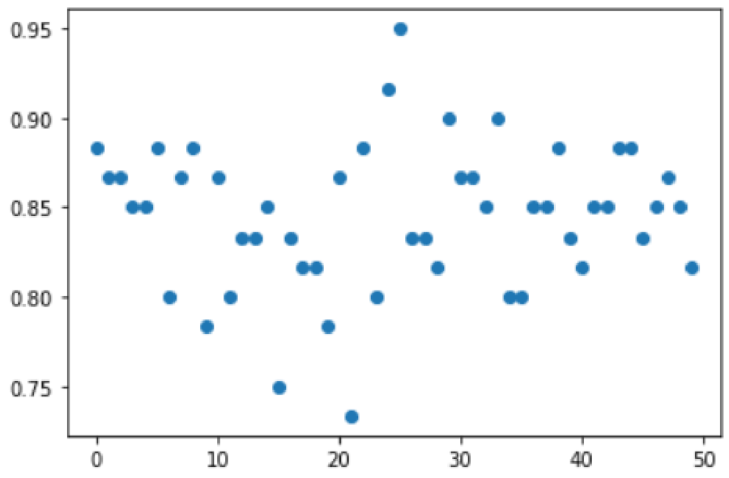
4.5. Feature Selection based on Pearson's Correlation Coefficient (Test Set)



	N means	Bayes	Perceptron	MSE	SVM Linear	SVM RBF
Accuracy	0.833	0.750	0.916	0.95	0.233	0.783
F1 Score	0.750	0.666	0.888	0.92	0.206	0.763

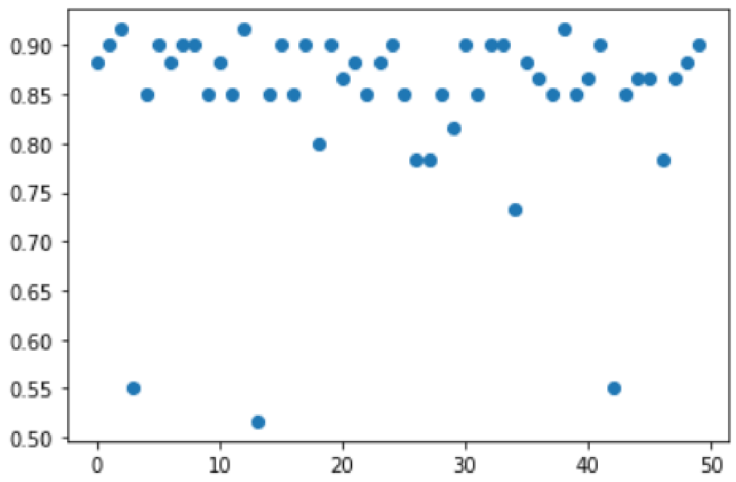
As mentioned in above section, I used Pearson's correlation to select the features from the expanded feature space. The above table consists of the accuracy and f1 scores for the minimum threshold value of 0.33 for all models. As we can see the MSE and perceptron models performed the best, but there is a catch as mentioned earlier the MSE and Perceptron model initialized weights randomly at start and it also randomly shuffles the dataset, this varies the accuracy and f1 score for each run.

MSE:



As the accuracies varied in the MSE model, I iterated through it 50 times capturing and plotting its accuracy for each run. As we can see the spread is huge ranging from 75% - 95%. I received an accuracy of 95% only once in 50 iteration and the avg hovered over 85%. So, as we can see this model is not reliable and we need to hardcode many parameters to get a 95% accuracy which will not generalize the model.

Perceptron:



I even iterated the perceptron model for 50 runs, this model had a lower spread and an avg accuracy of 88%-90% most of the times give the highest at 91.66%. This model is more reliable than the MSE and can be used.

Out of all the models using Pearson's correlation feature selection I would choose the perceptron model.

4.6. Feature Selection based on Accuracy of individual features (Test Set)

	N means	Bayes	Perceptron	MSE	SVM Linear	SVM RBF
Accuracy	0.833	0.883	0.800	0.616	0.500	0.783
F1 Score	0.750	0.844	0.749	0.000	0.399	0.711

As the best models using Pearson's correlation were unreliable, I used a different method of feature selection based on accuracy of each model trained only one features and selecting features above a threshold accuracy. For the models above I choose a threshold accuracy of 65% as it worked the best for me. The best model was the bayes model with the accuracy of 88.33% and F1 score of 0.844. But this model was more reliable and gave same accuracy constantly.

4.7. Normalizing Selected features for best models

Acc refers to Accuracy based feature selected data

Pea refers to Pearson's Correlation based feature selected data

	N means Acc	Bayes Acc	Perceptron Acc	N means Pea	Perceptron Pea
Accuracy	0.816	0.800	0.850	0.716	0.716
F1 Score	0.755	0.777	0.823	0.701	0.730

I tried to normalize the features and test them on the best models I had obtained, the accuracies dropped on average of 5-10% on all models. So, I excluded the normalizing process for the final models.

4.8. Standardizing Selected features for best models

Acc refers to Accuracy based feature selected data

Pea refers to Pearson's Correlation based feature selected data

	N means Acc	Bayes Acc	Perceptron Acc	N means Pea	Perceptron Pea
Accuracy	0.850	0.800	0.766	0.733	0.733
F1 Score	0.808	0.793	0.766	0.714	0.733

I tried to standardize the features and test them on the best models I had obtained, the accuracies dropped on average of 10-15% on all models, except the n-means acc. So, I excluded the normalizing process for the final models.

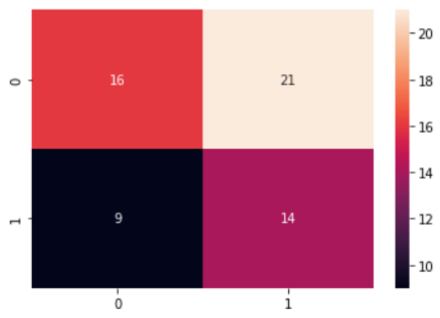
4.9. Best Models

4.9.1. Perceptron Learning Algorithm with sequential gradient decent using a scheduler for eta = a/(b+i) for values a = [0.01,0.1,1,10,100] and b = [1,10,100,1000], trained over the feature selected by the Pearson's correlation coefficient with a minimum threshold of 0.33 or 0.5.

The trivial, n-means and perceptron models were trained on the person's correlation-based features selected

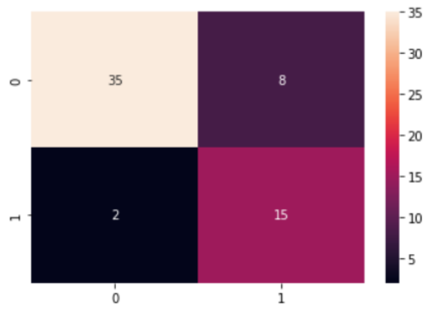
Trivial:

```
Test Set Accuracy = 0.5
 F1 Score = 0.48275862068965525

Confusion Matrix
```



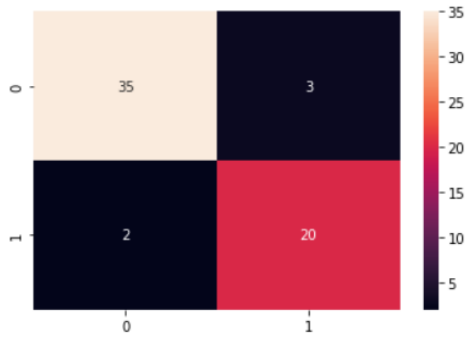
N-Means:

```
Test Set Accuracy = 0.8333333333333334
 F1 Score = 0.75

Confusion Matrix
```



Perceptron:

```
Accuracy = 0.9166666666666666
F1 Score = 0.888888888888889

Confusion Matrix
```

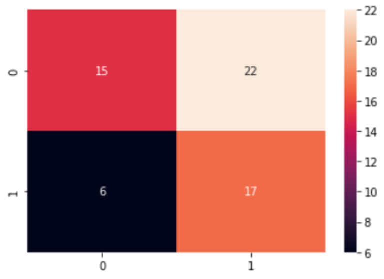


4.9.2. Bayes Minimum Error Classifier – Gaussian Distribution model, used on the accuracy-based feature selected data with a minimum threshold of 65%.

The trivial, n-means and Bayes models were trained on the accuracy-based features selected.

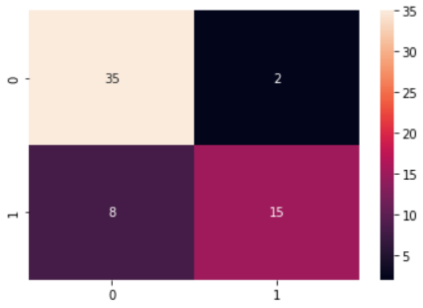
Trivial:

```
Test Set Accuracy = 0.5333333333333333
 F1 Score = 0.5483870967741935

Confusion Matrix
```



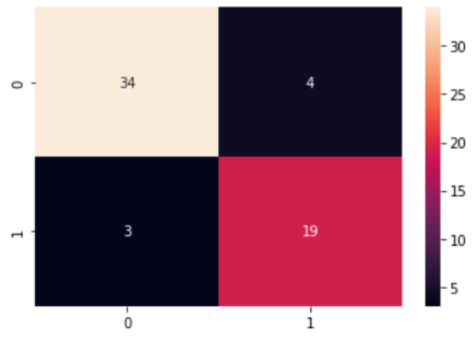
N-Means:

```
Test Set Accuracy = 0.8333333333333334
 F1 Score = 0.75

Confusion Matrix
```



Bayes:

```
Test Set Accuracy = 0.8833333333333333
 F1 Score = 0.8444444444444444

Confusion Matrix
```



Although the Bayes model's accuracy was lower than the best Perceptron model's accuracy, it gave me the same accuracy every time as there were no random elements involved in the model and was less computationally expensive as no extra parameters like eta was involved.

So, the Bayes model was the best model for me due to its reliability.

5. Libraries used and what you coded yourself

5.1. Libraries Used

- Numpy
- Pandas
- Matplotlib
- Random
- Scipy – used to calculate Euclidian distance
- Sklearn – Used to calculate f1 score
- Seaborn – Used to display confusion matrix clearly
- Math – Used for calculating log and exp
- Datetime – Used for calculating previous dates
- Warnings – remove unwanted warnings

5.2. Coded Myself

- Accuracy(y,yh) – gives the accuracy of predicted labels compared to true labels
- scores(yt,yh,sets) – gives the accuracy,f1,score,confusion matrix for predicted output, sets is a string value just to say if its test or validation.

- class_ind(X,y) – gives the indices of datapoints belonging to a particular class.
- trivial(l,df) – takes in data-frame and length of test set, to give randomly generated outputs based probability of class.
- nmeans(X,y,data) – takes in train datapoints, train labels, and test datapoints and give output labels.
- Bayes(df,X,y,Xt) – takes in data frame, train datapoints, train labels, and test datapoints and gives output labels.
- critierion_perc(w,z,x) – takes in weights, reflection coefficient (z) and datapoints and gives out cost J(w) for perceptron.
- gradient_seq(df) – takes in dataframe and gives out the optimal weights and its corresponding error for perceptron.
- classification_perc(w,x) – takes in optimal weights and test datapoints and gives out the predicted labels for perceptron.
- Perceptron(dfs,xt) – takes in the numpy converted dataframe and test datapoints, performs all necessary helper function mentioned as _perc above and gives the predicted labels.
- criterion_mse(w,x,y) - takes in weights, reflection coefficient (z) and datapoints and gives out cost J(w) for MSE.
- gradient_mse(df) - takes in dataframe and gives out the optimal weights and its corresponding error for MSE.
- classification_mse(w,x) - takes in optimal weights and test datapoints and gives out the predicted labels for MSE.
- MSE(dfs,xt) - takes in the numpy converted dataframe and test datapoints, performs all necessary helper function mentioned as _mse above and gives the predicted labels.
- get_z(y) – gives us the relflection coefficient 1 for class 0 and -1 for class 1.
- SVM_linear(X,z,gamma,Xt) – takes in train datapoints, reflection coefficient, gamma value is set to None as its not used in this and test datapoints and gives out the predicted labels.
- SVM_RBF(X,z,gamma,Xt) - takes in train datapoints, reflection coefficient, gamma value and test datapoints and gives out the predicted labels.
- Cross_Validation(model,name,df) – takes in the model, the models name and the dataframe and prints out each epochs error and return the epoch number with the highest validation set accuracy.
- low_epoch(i,model,name,df,Xt,yt)- takes in the epoch number with highest accuracy, model, model's name, dataframe, test datapoints, true labels and gives out the scores i.e. accuracy, f1 score and confusion matrix.
- fea_exp(dfex) – takes in data frame and expands the features in it.

- Cross_Validation_expand(model,name,df) - takes in the model, the models name and the expanded dataframe and prints out each epochs error and return the epoch number with the highest validation set accuracy.
- low_epoch_expand(i,model,name,df,Xt,yt) - takes in the epoch number with highest accuracy, model, model's name, expanded dataframe, expanded test datapoints, true labels and gives out the scores i.e. accuracy, f1 score and confusion matrix.
- pcc_fea_sel(dfex,val) – takes in data frame and a desired threshold value and return a list of features have a correlation value more than the threshold.
- fet_sel_acc(df,df_t,model,name,desired_acc) – takes in train dataframe, test dataframe, model to be used, model's name, desired threshold accuracy required, gives out a list of feature having a accuracy more than the threshold.
- acc_feat_sel(feat,df,df_t,model,name) = takes in the selected features, train dataframe, test dataframe, model to be used and model,s name and give the accuracy,f1 score and confusion matrix.

6. Summary and conclusions

After using different models, pre-processing, feature engineering, feature dimensionality reduction, parameter tunning, analysis and interpretations, the Bayes model was the best fit I could find for this data set. Though I think with a bit more work on the perceptron model I can achieve a much higher accuracy and F1 score, which I will consider as a task or follow up work after the project. Besides that, I think I am at a much better position in understanding ML models, creating them and trouble shooting various problems that arise, understand the complications and behind the scenes working of the models, after this project and EE559 course as in general.

References

[1] Troubleshooting - https://stackoverflow.com/

[2] Documentations - https://docs.python.org/3/, https://numpy.org/doc/, https://pandas.pydata.org/pandas-docs/stable/

[3] Classwork Notes, Homework Problems