Taha Afzal

Dr. Steinwand

COSC 380

16 May 2020

Rain in Australia

ABSTRACT

The goal for the project for me was to explore a dataset obtained from Kaggle about rain in Australia and then come up with a model with good accuracy. Using data such as the temperature, cloud cover, humidity, whether it rained that day or not, direction and speed of wind, my goal was to predict whether it will rain the following day or not- specifically the RainTomorrow variable with 0 being it will not rain and 1 being it will rain.

To achieve the goal, I explored the data first and noticed the relationships between the variables in the dataset after I had established basic understanding of the data. Clearly understanding the data dictionary and taking help of resources available online to get domain knowledge was quite helpful to understand why certain relationships between data exist and how they might be correlated with each other. For visualizations in the data exploration stage, I used the seaborn package. As there were quite a few categories in the categorical columns, it significantly increased the dimensions of the dataset. It was a tough choice to keep the variables even though they might not be correlated to the outcome variable and have Principal Component Analysis (PCA) reduce the dimensions itself.

Realizing that the RISK_MM variable might leak results to my models also proved to be very important and helpful and avoided the model from overfitting the data. In this version of the notebook, I tried some models using techniques not used in the previous versions so I also compare and contrast how the results are different now. Some of the changes include imputing missing data values or getting rid of the rows with missing values and manually removing variables to avoid overfitting or utilize PCA for that job.

DATA

Introduction

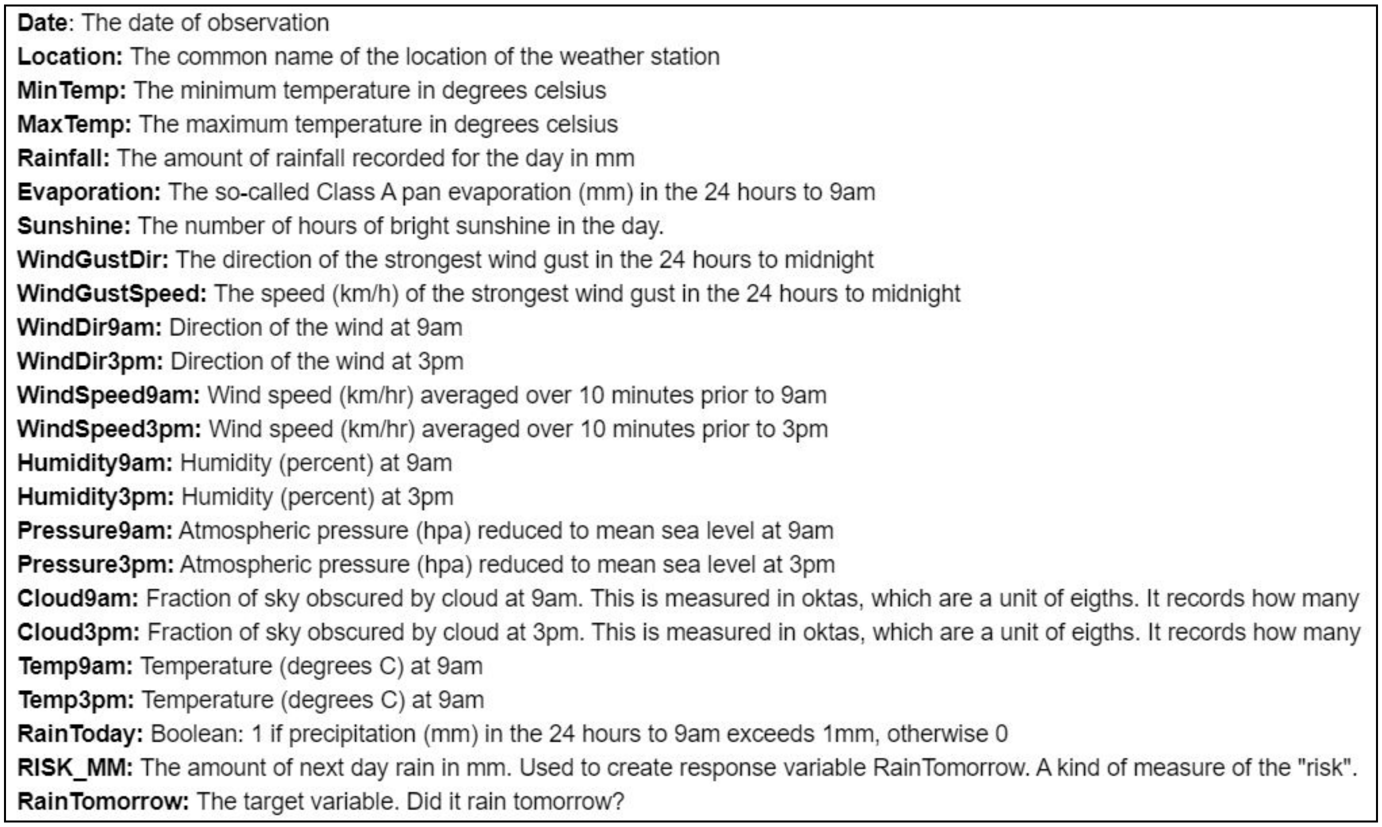
The dataset I used for this project was obtained from Kaggle. It is laid out as 24 columns in a Comma-separated values (CSV) file. The dataset contains (almost) daily weather observations and recordings from multiple weather stations all across Australia for just less than 10 years. This includes data from 49 weather stations out of a total of 870 weather stations in Australia, as mentioned by fishranger.com.

Deeper view

The earliest recording in the dataset is of 2007-11-01 whereas the latest is of 2017-06-25. This means the data is spread out over almost 10 years. This makes about 3524 calendar days but the unique dates in the dataset are only 3436, which means that some days were skipped and no recording was made by any of the weather stations on those certain days. It should be noted that the data includes all seasons so the trend in the data would not be similar all across the dataset. One more important thing to be aware of is that some spells of rain may exceed a day.

The RISK_MM and Rainfall variables record the amount of precipitation that reaches the ground, such as rain, drizzle, hail and snow depending on the weather. Throughout the paper, when I refer to rain, it would imply precipitation in any of the above mentioned forms.

The data dictionary included in Figure 1 shows a closer view of what variables are included in the dataset and what they mean. We have metrics about the weather including, but not limited to, the minimum and maximum temperatures during the day, wind speed's direction, wind direction, wind speed, temperature, humidity, pressure, and cloud at both 9 am and 3 pm the same day. When needed, the Metric System of units is used with the data observations.

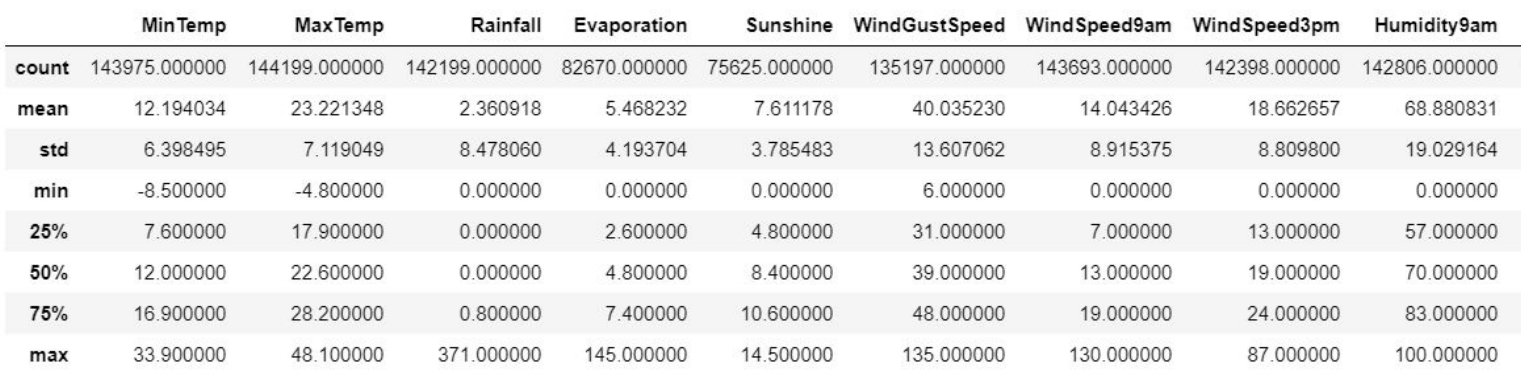


Date:	The date of observation
Location:	The common name of the location of the weather station
MinTemp:	The minimum temperature in degrees celsius
MaxTemp:	The maximum temperature in degrees celsius
Rainfall:	The amount of rainfall recorded for the day in mm
Evaporation:	The so-called Class A pan evaporation (mm) in the 24 hours to 9am
Sunshine:	The number of hours of bright sunshine in the day.
WindGustDir:	The direction of the strongest wind gust in the 24 hours to midnight
WindGustSpeed:	The speed (km/h) of the strongest wind gust in the 24 hours to midnight
WindDir9am:	Direction of the wind at 9am
WindDir3pm:	Direction of the wind at 3pm
WindSpeed9am:	Wind speed (km/hr) averaged over 10 minutes prior to 9am
WindSpeed3pm:	Wind speed (km/hr) averaged over 10 minutes prior to 3pm
Humidity9am:	Humidity (percent) at 9am
Humidity3pm:	Humidity (percent) at 3pm
Pressure9am:	Atmospheric pressure (hpa) reduced to mean sea level at 9am
Pressure3pm:	Atmospheric pressure (hpa) reduced to mean sea level at 3pm
Cloud9am:	Fraction of sky obscured by cloud at 9am. This is measured in oktas, which are a unit of eigths. It records how many
Cloud3pm:	Fraction of sky obscured by cloud at 3pm. This is measured in oktas, which are a unit of eigths. It records how many
Temp9am:	Temperature (degrees C) at 9am
Temp3pm:	Temperature (degrees C) at 9am
RainToday:	Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0
RISK_MM:	The amount of next day rain in mm. Used to create response variable RainTomorrow. A kind of measure of the "risk".
RainTomorrow:	The target variable. Did it rain tomorrow?

Figure 1: Data Dictionary

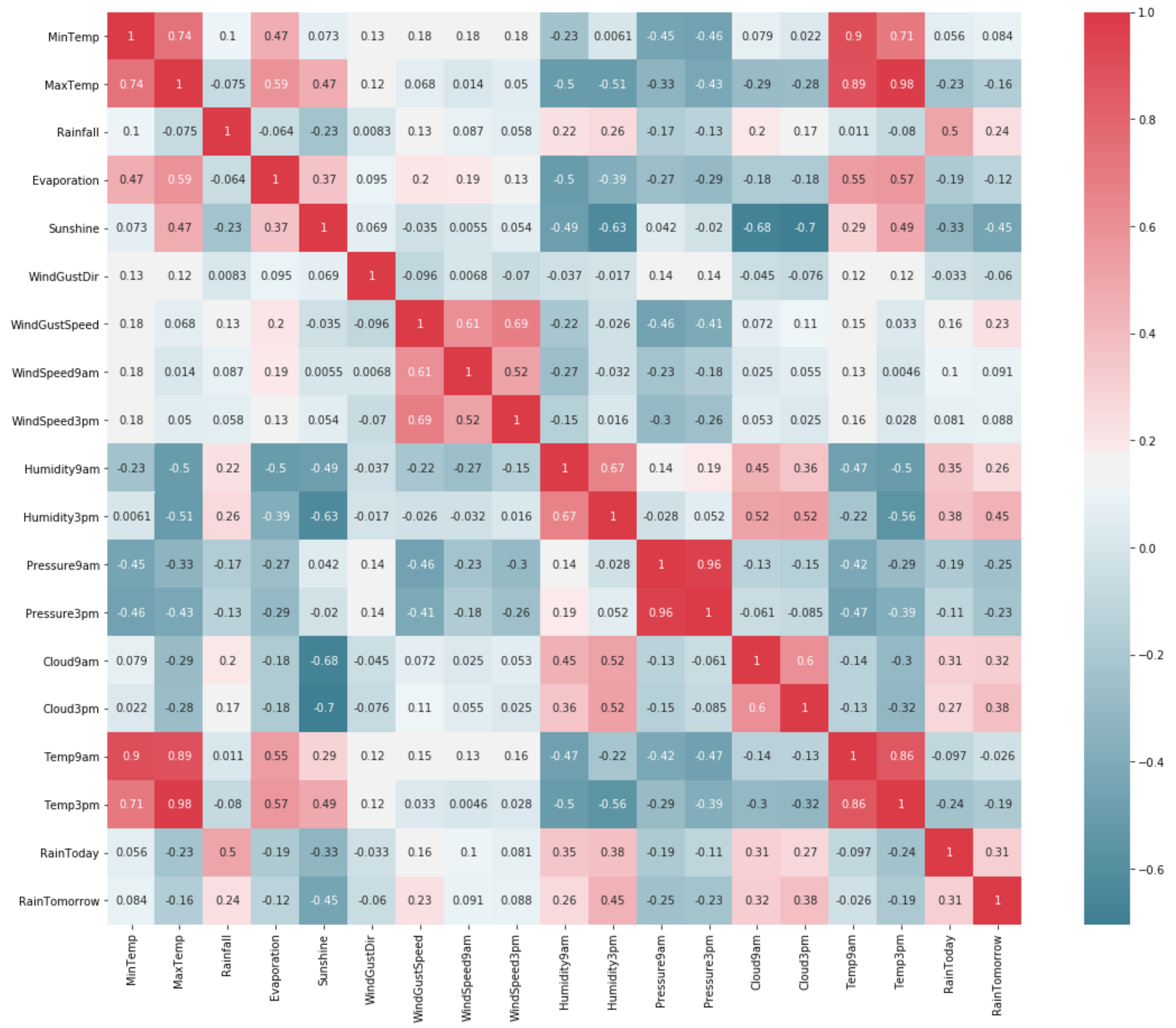
Data Exploration

Figure 2 includes the results produced by the describe() method on the dataframe showing the minimum, maximum, mean, standard deviation of the numerical fields. According to the BBC, Sydney received record-breaking rain in 30 years when it rained about 391.6 mm on the 4 days before 10 February 2020. This amount of rain is slightly higher than the maximum rainfall recorded in our dataset, which is 371 mm. In our dataset, some of the days saw no sunshine, some were without any wind, so there is a wide range of combinations represented in the dataset.



	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am
count	143975.000000	144199.000000	142199.000000	82670.000000	75625.000000	135197.000000	143693.000000	142398.000000	142806.000000
mean	12.194034	23.221348	2.360918	5.468232	7.611178	40.035230	14.043426	18.662657	68.880831
std	6.398495	7.119049	8.478060	4.193704	3.785483	13.607062	8.915375	8.809800	19.029164
min	-8.500000	-4.800000	0.000000	0.000000	0.000000	6.000000	0.000000	0.000000	0.000000
25%	7.600000	17.900000	0.000000	2.600000	4.800000	31.000000	7.000000	13.000000	57.000000
50%	12.000000	22.600000	0.000000	4.800000	8.400000	39.000000	13.000000	19.000000	70.000000
75%	16.900000	28.200000	0.800000	7.400000	10.600000	48.000000	19.000000	24.000000	83.000000
max	33.900000	48.100000	371.000000	145.000000	14.500000	135.000000	130.000000	87.000000	100.000000

Figure 2: df.describe()



	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
MinTemp	1	0.74	0.1	0.47	0.073	0.13	0.18	0.18	0.18	-0.23	0.0061	-0.45	-0.46	0.079	0.022	0.9	0.71	0.056	0.084
MaxTemp	0.74	1	-0.075	0.59	0.47	0.12	0.068	0.014	0.05	-0.5	-0.51	-0.33	-0.43	-0.29	-0.28	0.89	0.98	-0.23	-0.16
Rainfall	0.1	-0.075	1	-0.064	-0.23	0.0083	0.13	0.087	0.058	0.22	0.26	-0.17	-0.13	0.2	0.17	0.011	-0.08	0.5	0.24
Evaporation	0.47	0.59	-0.064	1	0.37	0.095	0.2	0.19	0.13	-0.5	-0.39	-0.27	-0.29	-0.18	-0.18	0.55	0.57	-0.19	-0.12
Sunshine	0.073	0.47	-0.23	0.37	1	0.069	-0.035	0.0055	0.054	-0.49	-0.63	0.042	-0.02	-0.68	-0.7	0.29	0.49	-0.33	-0.45
WindGustDir	0.13	0.12	0.0083	0.095	0.069	1	-0.096	0.0068	-0.07	-0.037	-0.017	0.14	0.14	-0.045	-0.076	0.12	0.12	-0.033	-0.06
WindGustSpeed	0.18	0.068	0.13	0.2	-0.035	-0.096	1	0.61	0.69	-0.22	-0.026	-0.46	-0.41	0.072	0.11	0.15	0.033	0.16	0.23
WindSpeed9am	0.18	0.014	0.087	0.19	0.0055	0.0068	0.61	1	0.52	-0.27	-0.032	-0.23	-0.18	0.025	0.055	0.13	0.0046	0.1	0.091
WindSpeed3pm	0.18	0.05	0.058	0.13	0.054	-0.07	0.69	0.52	1	-0.15	0.016	-0.3	-0.26	0.053	0.025	0.16	0.028	0.081	0.088
Humidity9am	-0.23	-0.5	0.22	-0.5	-0.49	-0.037	-0.22	-0.27	-0.15	1	0.67	0.14	0.19	0.45	0.36	-0.47	-0.5	0.35	0.26
Humidity3pm	0.0061	-0.51	0.26	-0.39	-0.63	-0.017	-0.026	-0.032	0.016	0.67	1	-0.028	0.052	0.52	0.52	-0.22	-0.56	0.38	0.45
Pressure9am	-0.45	-0.33	-0.17	-0.27	0.042	0.14	-0.46	-0.23	-0.3	0.14	-0.028	1	0.96	-0.13	-0.15	-0.42	-0.29	-0.19	-0.25
Pressure3pm	-0.46	-0.43	-0.13	-0.29	-0.02	0.14	-0.41	-0.18	-0.26	0.19	0.052	0.96	1	-0.061	-0.085	-0.47	-0.39	-0.11	-0.23
Cloud9am	0.079	-0.29	0.2	-0.18	-0.68	-0.045	0.072	0.025	0.053	0.45	0.52	-0.13	-0.061	1	0.6	-0.14	-0.3	0.31	0.32
Cloud3pm	0.022	-0.28	0.17	-0.18	-0.7	-0.076	0.11	0.055	0.025	0.36	0.52	-0.15	-0.085	0.6	1	-0.13	-0.32	0.27	0.38
Temp9am	0.9	0.89	0.011	0.55	0.29	0.12	0.15	0.13	0.16	-0.47	-0.22	-0.42	-0.47	-0.14	-0.13	1	0.86	-0.097	-0.026
Temp3pm	0.71	0.98	-0.08	0.57	0.49	0.12	0.033	0.0046	0.028	-0.5	-0.56	-0.29	-0.39	-0.3	-0.32	0.86	1	-0.24	-0.19
RainToday	0.056	-0.23	0.5	-0.19	-0.33	-0.033	0.16	0.1	0.081	0.35	0.38	-0.19	-0.11	0.31	0.27	-0.097	-0.24	1	0.31
RainTomorrow	0.084	-0.16	0.24	-0.12	-0.45	-0.06	0.23	0.091	0.088	0.26	0.45	-0.25	-0.23	0.32	0.38	-0.026	-0.19	0.31	1

Figure 3: Heatmap using the seaborn package

Heatmap

After that, I plotted a heatmap for the variables to see if and how much are they correlated to each other and the outcome variable. This heatmap is represented in Figure 3 above. From the heatmap, we can derive the following observations:

1. The highest positive correlation with RainTomorrow is of Humidity3pm of 0.45. This is understandable as according to smartfog.com, it rains when humidity is at 100% which is when the cloud cannot retain any more water. So, this makes sense as the higher the humidity, the more chances of rain the next day. This is followed by 0.38 with Cloud3pm, 0.32 with Cloud9am, and 0.31 with RainToday. The correlation with Cloud3pm and Cloud9am also makes sense as the presence of clouds means a chance of rain. A fair amount of correlation can also be observed between Cloud9am, Cloud3pm and Humidity9am and Humidity3pm as according to Drew, Misha, Jenna M., Jenna O., Courtney, and Jessica (windows2universe), more humidity leads to the formation of more clouds.
2. The highest negative correlation with RainTomorrow is of Sunshine of -0.45. Sunshine is positively correlated with MaxTemp and Temp3pm. The article from Ideo.columbia.edu explains the phenomenon about the relation between higher temperatures and low rainfall quite well. Due to contamination from man-made aerosols, despite rising temperatures from more sun rays, rainfall is decreasing. As opposed to our intuition that more sunshine leads to more evaporation forming more clouds and hence resulting in more rain, the negative correlation between Sunshine and RainTomorrow is explained by this article. It should also be noted that if we see cloud cover in relation with sunshine, they are strongly negatively correlated meaning summer season with less rainfall compared to a rainy season, which is usually between November and April in Australia according to weatheronline.

Having insights into these above mentioned correlations is helpful and will help us better understand why the variables are related in the way that they are.

Pairplot

From the pairplot from the seaborn package, (equivalent to scatterplot in matplotlib.pyplot), shown in Figure 4, it can be observed that the fields representing temperature are related to each other- especially MaxTemp and Temp3pm as the temperature would be the highest during peak day. Apart from this, the variables representing pressure (Pressure9am and Pressure3pm) were also strongly correlated with each other.

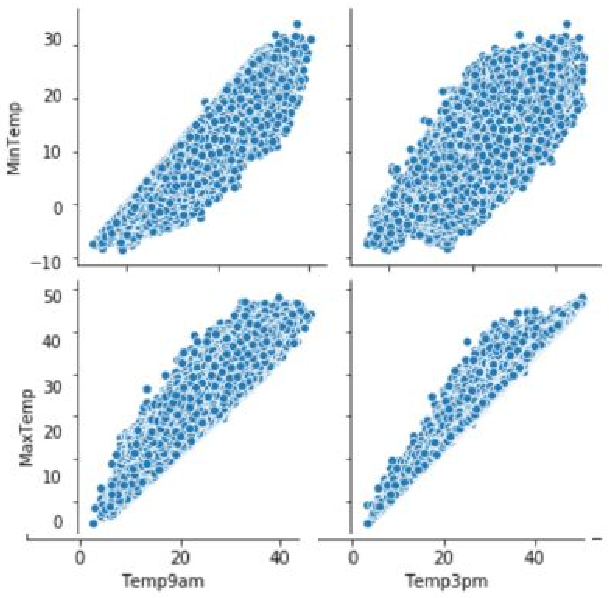


Figure 4: Pairplot

Data Cleaning and Preparation

To make things simple, I decided to see Australia as a whole when trying to predict if it will rain in Australia, in general, the next day or not. This can be done by removing the Date and Location columns from the dataset. When using RainTomorrow as the target variable, it should be taken into account that we should also remove the RISK_MM column from the dataset as it might "leak" the results to the models I train. RISK_MM includes the amount of rainfall in millimeters for the next day. If the RISK_MM variable is greater than 0, RainTomorrow would be set to Yes as RISK_MM is used as a response variable according to the data dictionary. I removed this field to avoid giving the model data from the future when it is trying to predict the results on unseen data.

The dataset had a lot of missing values. To overcome this, I tried two approaches: either to impute the data or to remove the rows with missing values. In another version of this project, I tried imputing these values and since the predictive accuracy of Logistic Regression and Support Vector Machine was not the best compared to when removing these rows, I decided to go with the latter option for this version of the project. Using GridSearch on the training and validation sets with imputed data did not show promising results either when compared to the latter approach.

The missing values also included numerical values represented as "NaN" which the default dropna method (without any parameters) could not catch. Reading more about the method's api from pandas.pydata, I was able to set the "how" parameter to "any" so that "NaN" values would also be considered as missing values and removed.

RainToday and RainTomorrow variable values were also converted to 0s and 1s by replacing the No and Yes respectively. This was easily done using the replace() method. The same technique was used to replace the values in the WindGustDir field to numbers. The numerical variables were also scaled between 0 and 1 using MinMaxScaler() from sklearn.preprocessing.

Preparation and Partition of Data

Converting categorical variables into dummy variables added many variables leading to a risk of the models over-fitting the dataset. Dummy variables were created for all the categories for the WindDir9am and WindDir3pm columns adding about 28 variables to the dataset!

After plotting the heatmap (Figure 3), instead of dropping predictors that did not have a significant correlation with RainTomorrow, I decided to run Principal Components Analysis (PCA) as a dimension reduction technique while maintaining 95% of the data variance. Reducing the dimensionality of the dataset based on correlation only, there was a huge risk of removing variables that might be affecting RainTomorrow indirectly and is not the best approach.

The dataset was finally partitioned into training, validation to adjust the model's hyperparameters, and test sets. 60% of the data was training, and 20% for validation and testing each.

MODELS

Logistic Regression

The logistic regression model accuracy score is 0.8345. So, the model performs well to predict if it will rain tomorrow in Australia or not. The number of predictions that it will rain the following day are quite less when compared to the observations predicting that it will not rain. Increasing the value of C results in higher test set accuracy and also an increased training set accuracy. Thus, it can be said that a more complex model will perform better.

I also increased the threshold level for the model when it is classifying based on the probabilities, which increased the accuracy of the model. ROC AUC of the model (shown in Figure 5) also approaches 1, which is another good sign. The AUC for the model was 85.61

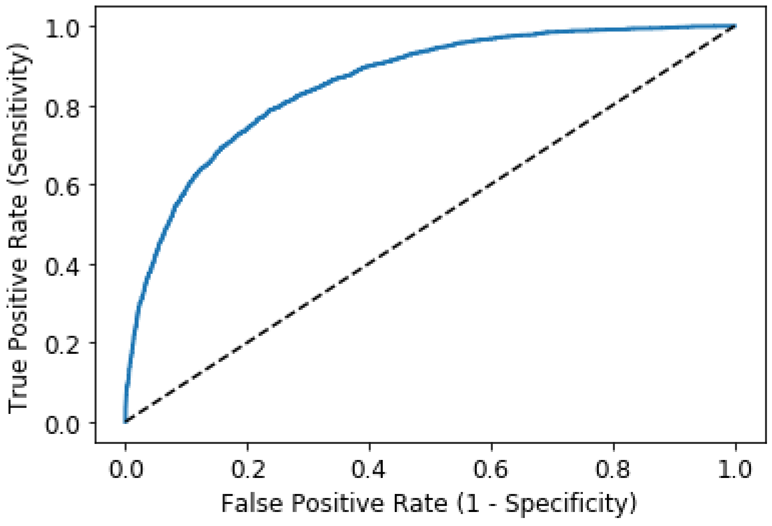
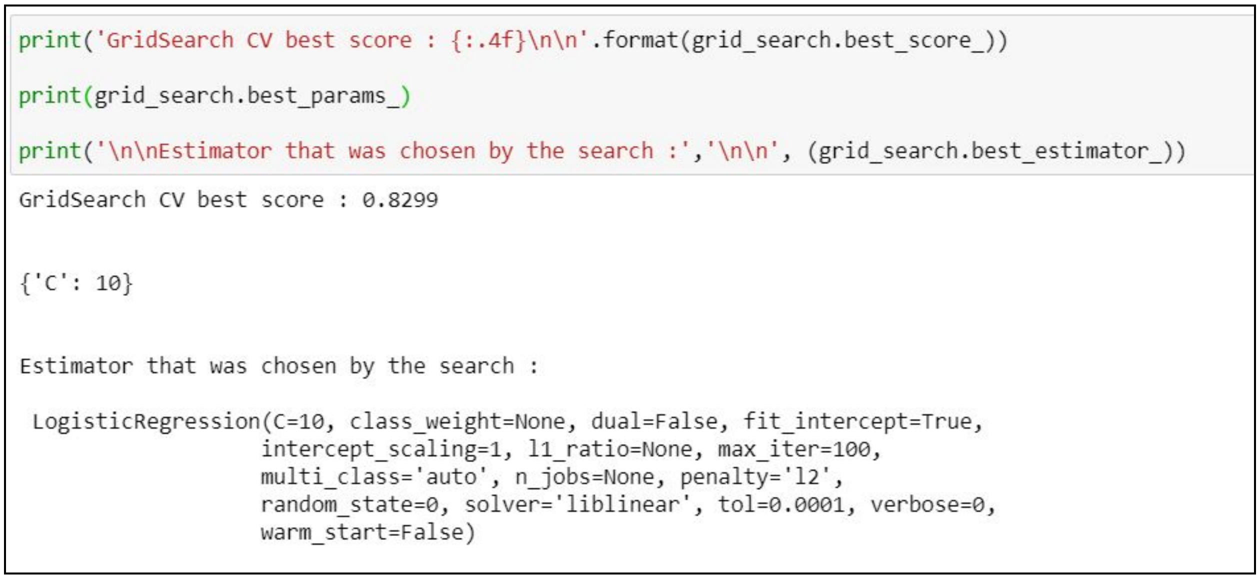


Figure 5: ROC

Cross-validation did not result in a significant higher performance of the model. My original model test accuracy was 0.8345 while GridSearchCV accuracy was 0.8347. So, it can be seen that GridSearchCV improved the performance for the model though it was not by a lot. The results obtained by the GridSearchCV are shown below in Figure 6. It identified the value of C to be 10 as the best parameter and the estimator as shown.



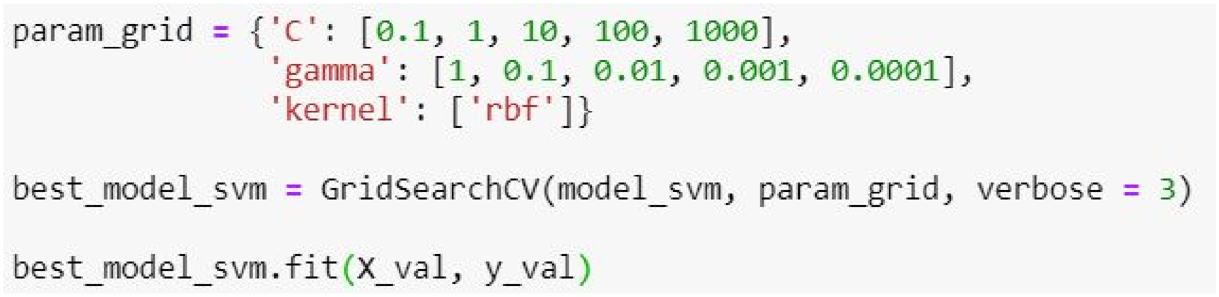
```
print('GridSearch CV best score : {:.4f}\n\n'.format(grid_search.best_score_))

print(grid_search.best_params_)

print('\n\nEstimator that was chosen by the search :','\n\n', (grid_search.best_estimator_))
```

```
GridSearch CV best score : 0.8299


{'C': 10}


Estimator that was chosen by the search :

 LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

Figure 6: GridSearchCV on logistic regression results

Support Vector Machine

The SVM with hyper-parameters found by GridSearchCV performed significantly well compared to my original SVM with the default parameters. The values for C, gamma, and kernel can be seen in Figure 7 below. The comparison is an increase from 82.2% accuracy on the test set to 86.5% on the test set. The accuracy of the model with the tuned hyper parameters can be seen in Figure 8. In comparison with the Logistic Regression, the SVM outperforms with noticeable margin.



```
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

best_model_svm = GridSearchCV(model_svm, param_grid, verbose = 3)

best_model_svm.fit(X_val, y_val)
```

Figure 7: GridSearchCV on SVM

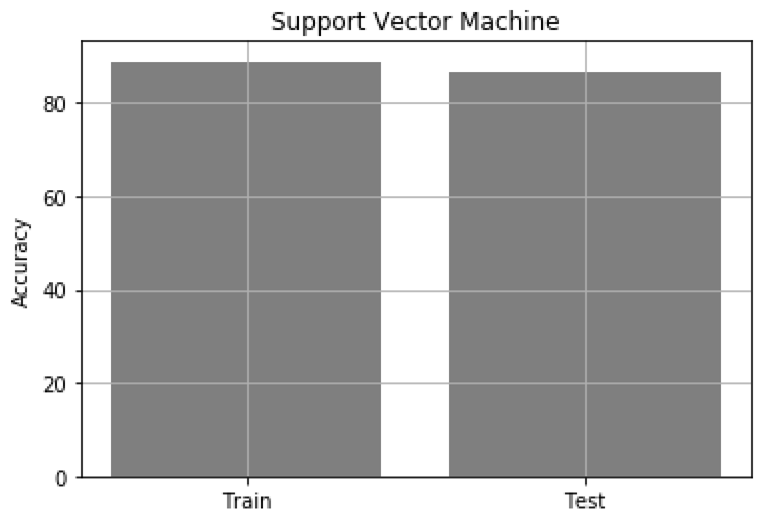


Figure 8: SVM Train vs Test

Gradient Boost Classifier

The Gradient Boost Classifier proved to be the least performing model on this dataset even after tuning the hyperparameters. With the default hyperparameters, it had an accuracy of 77.3% on the training set and 76.3% on the test set. Using GridSearchCV did increase the accuracy to 79.4% on the training set and 78.2% on the test (as shown in Figure 9) but it is still far away from the logistic regression and the support vector machine.

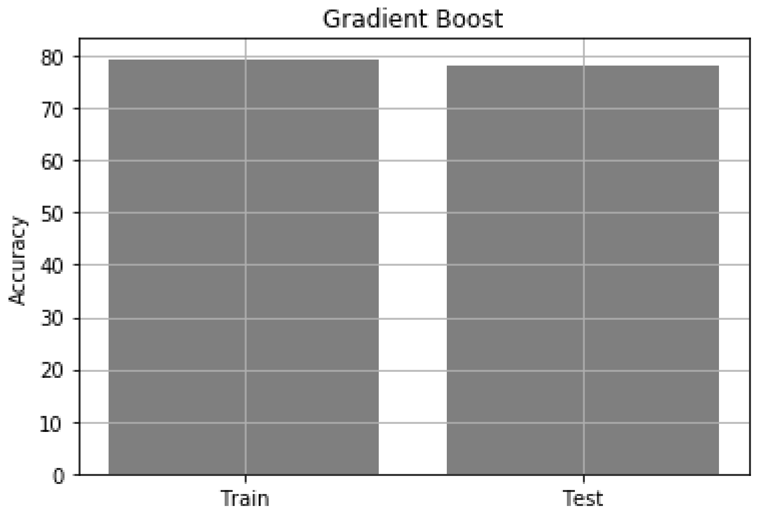


Figure 9: Gradient Boost Train vs Test

CONCLUSION

In my opinion, and as the results show, the best model among the three I trained, the best model for this task would be the Support Vector Machine with the hyperparameters adjusted as suggested by the GridSearchCV resulting in 86.5% accuracy on the test set. GridSearchCV proved to be helpful for all the models above, but it varied quite a bit on how better the model could be from its default state once it’s hyperparameters are tuned. Compared to the other versions of the notebook, where I imputed the missing numerical data, all of these models did not perform as well. To sum up, if I use the SVM to predict if it will rain the following day in Australia as a whole, based on the variables provided, it has a 86.5% chance to get it right.

Works Cited

Géron Aurélien. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. OReilly Media, Inc., 2019.

Brownlee, Jason. "How to Prepare Text Data for Machine Learning with Scikit-Learn." Machine Learning Mastery, 7 Aug. 2019, machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/.

Young, Joe. "Rain in Australia." Kaggle, 3 Dec. 2018, www.kaggle.com/jsphyg/weather-dataset-rattle-package.

Young, Joe. "Rain in Australia." *Kaggle*, 3 Dec. 2018, www.kaggle.com/jsphyg/weather-dataset-rattle-package/discussion/78316.

"The Truth about Weather Sites." The Truth about Weather Sites, www.fishranger.com.au/the_truth_about_weather_sites.

"Sydney Rains: Record Rainfall Brings Flooding but Puts out Mega-Blaze." *BBC News*, BBC, 10 Feb. 2020, www.bbc.com/news/world-australia-51439175.

"How Rain and Humidity Connected?" *Smart Fog*, 24 Jan. 2020, www.smartfog.com/how-rain-and-humidity-connected.html.

"What Happens to the Weather When the Humidity Goes up? How Are Temperature and Humidity Related? Do Relative Humidity and Cloud Coverage Affect Each Other at All?" *What Happens to the Weather When the Humidity Goes up? - Windows to the Universe*, www.windows2universe.org/kids_space/humidity.html.

"Could Global Warming Mean Less Sunshine and Less Rainfall?" *Could Global Warming Mean Less Sunshine and Less Rainfall? | Lamont-Doherty Earth Observatory*, www.ldeo.columbia.edu/news-events/could-global-warming-mean-less-sunshine-and-les s-rainfall.

Weatheronline.co.uk. "Australia." *WeatherOnline*, www.weatheronline.co.uk/reports/climate/Australia.htm.

"Pandas.DataFrame.dropna." *Pandas.DataFrame.dropna - Pandas 1.0.3 Documentation*, pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html.

Appendix for the code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier

import seaborn as sns

df = pd.read_csv(r'D:\Augustana University\Courses\5. Spring 2020\COSC 380-A Artificial
Intelligence & Robotics\Projects\Final Project\weatherAUS.csv')
```

```
In [ ]:
df.shape
```

```
In [ ]:
df.head(10)
```

Date: The date of observation

Location: The common name of the location of the weather station

MinTemp: The minimum temperature in degrees celsius

MaxTemp: The maximum temperature in degrees celsius

Rainfall: The amount of rainfall recorded for the day in mm

Evaporation: The so-called Class A pan evaporation (mm) in the 24 hours to 9am

Sunshine: The number of hours of bright sunshine in the day.

WindGustDir: The direction of the strongest wind gust in the 24 hours to midnight

WindGustSpeed: The speed (km/h) of the strongest wind gust in the 24 hours to midnight

WindDir9am: Direction of the wind at 9am

WindDir3pm: Direction of the wind at 3pm

WindSpeed9am: Wind speed (km/hr) averaged over 10 minutes prior to 9am

WindSpeed3pm: Wind speed (km/hr) averaged over 10 minutes prior to 3pm

Humidity9am: Humidity (percent) at 9am

Humidity3pm: Humidity (percent) at 3pm

Pressure9am: Atmospheric pressure (hpa) reduced to mean sea level at 9am

Pressure3pm: Atmospheric pressure (hpa) reduced to mean sea level at 3pm

Cloud9am: Fraction of sky obscured by cloud at 9am. This is measured in oktas, which are a unit of eigths. It records how many

Cloud3pm: Fraction of sky obscured by cloud at 3pm. This is measured in oktas, which are a unit of eigths. It records how many

Temp9am: Temperature (degrees C) at 9am

Temp3pm: Temperature (degrees C) at 9am

RainToday: Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0

RISK_MM: The amount of next day rain in mm. Used to create response variable RainTomorrow. A kind of measure of the "risk".

RainTomorrow: The target variable. Did it rain tomorrow?

```
In [ ]:
df.describe()
```

```
In [ ]:
df.drop(labels = ['Date', 'Location', 'RISK_MM'], axis = 1, inplace = True)
```

```
In [ ]:
# Replacing Yes by 1 and No by 0 for RainToday and RainTomorrow columns

df['RainToday'].replace({'No': 0, 'Yes': 1}, inplace = True)
df['RainTomorrow'].replace({'No': 0, 'Yes': 1}, inplace = True)
```

```
In [ ]:
df['WindGustDir'].unique()
```

In [ ]:

```
df['WindGustDir'].replace({'W': 1, 'WNW': 2, 'WSW': 3, 'NE': 4, 'NNW':5, 'N':6, 'NNE': 7, 'SW': 8,
                          'ENE':9, 'SSE': 10, 'S': 11, 'NW': 12, 'SE':13, 'ESE':14,
                          'E': 15, 'SSW':16}, inplace = True)
```

In [ ]:

```
import seaborn as sns

f, ax = plt.subplots(figsize=(20, 15))
corr = df.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10,
as_cmap=True),
         square=True, ax=ax, annot = True)
```

In [ ]:

```
sns.pairplot(df)
```

In [ ]:

```
categorical = ['WindDir3pm', 'WindDir9am']
```

In [ ]:

```
# Adding dummy variables for categorical predictors

df = pd.get_dummies(df, columns = categorical, drop_first = True)
```

In [ ]:

```
# Getting rid of nan values

df = df.dropna(how = 'any')
df.shape
```

```
In [ ]:
from sklearn import preprocessing

scale = preprocessing.MinMaxScaler()
scale.fit(df)

df = pd.DataFrame(scale.transform(df), index = df.index, columns = df.columns)
```

```
In [ ]:
X = df.drop(labels = ['RainTomorrow'], axis = 1)
X
```

```
In [ ]:
y = df['RainTomorrow']
y
```

```
In [ ]:
# Splitting the dataset
from sklearn.model_selection import train_test_split

X_train_val, X_test, y_train_val, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val, test_size=0.2,
random_state=42)
```

```
In [ ]:
X_train.shape
```

```
In [ ]:
X_val.shape
```

In [ ]:

```
X_test.shape
```

In [ ]:

```
# Dimension Reduction to keep 95% of the variance. Reducing 16 variables!

from sklearn.decomposition import PCA

pca = PCA(n_components = 0.95)
X_train = pca.fit_transform(X_train)

X_val = pca.transform(X_val)
X_test = pca.transform(X_test)
```

In [ ]:

```
X_train.shape
```

In [ ]:

```
X_val.shape
```

In [ ]:

```
X_test.shape
```

Models

Logistic Regression

In [ ]:

```
from sklearn.linear_model import LogisticRegression
```

```
logreg = LogisticRegression(solver='liblinear', random_state=0)

logreg.fit(X_train, y_train)
```

```
In [ ]:
y_pred_test = logreg.predict(X_test)

y_pred_test
```

```
In [ ]:
# probability of no rain (0)

logreg.predict_proba(X_test)[:,0]
```

```
In [ ]:
# probability of rain (1)

logreg.predict_proba(X_test)[:,1]
```

```
In [ ]:
from sklearn.metrics import accuracy_score

print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred_test)))
```

```
In [ ]:
y_pred_train = logreg.predict(X_train)

print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train)))
```

```
In [ ]:
print('Training set score: {:.4f}'.format(logreg.score(X_train, y_train)))
print('Test set score: {:.4f}'.format(logreg.score(X_test, y_test)))

# Both values are comparable so, no overfitting.
```

In [ ]:

```
logreg100 = LogisticRegression(C=100, solver='liblinear', random_state=0)

logreg100.fit(X_train, y_train)
```

In [ ]:

```
print('Training set score: {:.4f}'.format(logreg100.score(X_train, y_train)))
print('Test set score: {:.4f}'.format(logreg100.score(X_test, y_test)))
```

In [ ]:

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred_test)

cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0'],
                         index=['Predict Positive:1', 'Predict Negative:0'])

sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

In [ ]:

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred_test))
```

In [ ]:

```
TP = cm[0,0]
TN = cm[1,1]
FP = cm[0,1]
```

```
FN = cm[1,0]

classification_accuracy = (TP + TN) / float(TP + TN + FP + FN)

print('Classification accuracy : {0:0.4f}'.format(classification_accuracy))
```

```
In [ ]:
```

```
classification_error = (FP + FN) / float(TP + TN + FP + FN)

print('Classification error : {0:0.4f}'.format(classification_error))
```

```
In [ ]:
```

```
precision = TP / float(TP + FP)

print('Precision : {0:0.4f}'.format(precision))
```

```
In [ ]:
```

```
recall = TP / float(TP + FN)

print('Recall or Sensitivity : {0:0.4f}'.format(recall))
```

```
In [ ]:
```

```
true_positive_rate = TP / float(TP + FN)

print('True Positive Rate : {0:0.4f}'.format(true_positive_rate))
```

```
In [ ]:
```

```
false_positive_rate = FP / float(FP + TN)

print('False Positive Rate : {0:0.4f}'.format(false_positive_rate))
```

In [ ]:

```
specificity = TN / (TN + FP)

print('Specificity : {0:0.4f}'.format(specificity))
```

In [ ]:

```
y_pred_prob = logreg.predict_proba(X_test)[0:10]

y_pred_prob
```

In [ ]:

```
logreg.predict_proba(X_test)[0:10, 1]
```

In [ ]:

```
y_pred1 = logreg.predict_proba(X_test)[:, 1]
```

In [ ]:

```
plt.rcParams['font.size'] = 12
plt.hist(y_pred1, bins = 10)
plt.title('Histogram of predicted probabilities of rain')
plt.xlim(0,1)
plt.xlabel('Predicted probabilities of rain')
plt.ylabel('Frequency')
```

In [ ]:

```
# plot ROC Curve

from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred1)
plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, linewidth=2)
plt.plot([0,1], [0,1], 'k--' )
plt.rcParams['font.size'] = 12
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.show()
```

```
In [ ]:
from sklearn.metrics import roc_auc_score

ROC_AUC = roc_auc_score(y_test, y_pred1)
print('ROC AUC : {:.4f}'.format(ROC_AUC))
```

```
In [ ]:
from sklearn.model_selection import cross_val_score

Cross_validated_ROC_AUC = cross_val_score(logreg, X_train, y_train, cv=5,
scoring='roc_auc').mean()
print('Cross validated ROC AUC : {:.4f}'.format(Cross_validated_ROC_AUC))
```

```
In [ ]:
from sklearn.model_selection import cross_val_score

scores = cross_val_score(logreg, X_train, y_train, cv = 5, scoring='accuracy')
print('Cross-validation scores:{}'.format(scores))
```

```
In [ ]:
print('Average cross-validation score: {:.4f}'.format(scores.mean()))
```

```
In [ ]:
from sklearn.model_selection import GridSearchCV

parameters = [{'penalty':['l1','l2']},
              {'C':[1, 10, 100, 1000]}]
```

```
grid_search = GridSearchCV(estimator = logreg,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 5,
                           verbose=0)
grid_search.fit(X_train, y_train)
```

In [ ]:

```
print('GridSearch CV best score : {:.4f}\n\n'.format(grid_search.best_score_))

print(grid_search.best_params_)

print('\n\nEstimator that was chosen by the search :','\n\n', (grid_search.best_estimator_))
```

In [ ]:

```
print('GridSearch CV score on test set: {0:0.4f}'.format(grid_search.score(X_test, y_test)))
```

Support Vector Machine

In [ ]:

```
from sklearn import svm

model_svm = clf_svc = svm.SVC()
model_svm.fit(X_train, y_train)
```

In [ ]:

```
X_train_pred = model_svm.predict(X_train)
confusion_matrix(y_train, X_train_pred)
```

In [ ]:

```
accuracy_svm_train = accuracy_score(y_train, X_train_pred)
print("accuracy on training set: ", accuracy_svm_train)
```

```
In [ ]:
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

best_model_svm = GridSearchCV(model_svm, param_grid, verbose = 3)

best_model_svm.fit(X_val, y_val)
```

```
In [ ]:
best_model_svm.best_params_
```

```
In [ ]:
X_val_pred = best_model_svm.predict(X_val)
confusion_matrix(y_val, X_val_pred)
```

```
In [ ]:
accuracy_svm_val = accuracy_score(y_val, X_val_pred)
print("best model's accuracy on validation set: ", accuracy_svm_val)
```

```
In [ ]:
X_test_pred = best_model_svm.predict(X_test)
confusion_matrix(y_test, X_test_pred)
```

```
In [ ]:
accuracy_svm_test = accuracy_score(y_test, X_test_pred)
print("accuracy on test set: ", accuracy_svm_test)
```

```
In [ ]:
models = ('Train', 'Test')
```

```
y_pos = np.arange(len(models))
accuracy = [accuracy_svm_train*100, accuracy_svm_test*100]

plt.bar(y_pos, accuracy, align='center', alpha=0.5, color = 'black')
plt.xticks(y_pos, models)
plt.ylabel('Accuracy')
plt.title('Support Vector Machine')
plt.grid(True)

plt.show()
```

Gradient Boost Classifier

In [ ]:

```
model_gb = GradientBoostingClassifier()
model_gb.fit(X_train, y_train)
```

In [ ]:

```
X_train_pred = model_gb.predict(X_train)
confusion_matrix(y_train, X_train_pred)
```

In [ ]:

```
accuracy_gb_train = accuracy_score(y_train, X_train_pred)
print("accuracy on training set: ", accuracy_gb_train)
```

In [ ]:

```
params = {'learning_rate': [0.001, 0.01, 0.1, 10, 100],
        'max_depth': [10, 100, 1000],
        'n_estimators': [1, 10, 100]
        }

best_model_gb = GridSearchCV(model_gb, params, verbose = 3)
```

```
best_model_gb.fit(X_val, y_val)
```

In [ ]:

```
best_model_gb.best_params_
```

In [ ]:

```
X_val_pred = best_model_gb.predict(X_val)
confusion_matrix(y_val, X_val_pred)
```

In [ ]:

```
accuracy_gb_val = accuracy_score(y_val, X_val_pred)
print("best model's accuracy on validation set: ", accuracy_gb_val)
```

In [ ]:

```
X_test_pred = best_model_gb.predict(X_test)
confusion_matrix(y_test, X_test_pred)
```

In [ ]:

```
accuracy_gb_test = accuracy_score(y_test, X_test_pred)
print("accuracy on test set: ", accuracy_gb_test)
```

In [ ]:

```
models = ('Train', 'Test')
y_pos = np.arange(len(models))
accuracy = [accuracy_gb_train*100, accuracy_gb_test*100]

plt.bar(y_pos, accuracy, align='center', alpha=0.5, color = 'black')
plt.xticks(y_pos, models)
plt.ylabel('Accuracy')
plt.title('Gradient Boost')
plt.grid(True)
```

```
plt.show()
```