```
#READING DATA
import pandas as pd
matches = pd.read_csv("matches.csv", index_col=0)
matches.head()
```

	date	time	comp	round	day	venue	result	gf	ga	opponent	...	match report
1	2021-08-15	16:30	Premier League	Matchweek 1	Sun	Away	L	0.0	1.0	Tottenham	...	Match Report
2	2021-08-21	15:00	Premier League	Matchweek 2	Sat	Home	W	5.0	0.0	Norwich City	...	Match Report
3	2021-08-28	12:30	Premier League	Matchweek 3	Sat	Home	W	5.0	0.0	Arsenal	...	Match Report
	2021-		Premier	Matchweek						Leicester		Match

```
matches.shape
```

```
(1389, 27)
```

```
#INVESTGATING MISSING DATA
#Should have 1520 matches in 2 seasons of EPL but here 1389 match data are present so let's figure out what's going on here
#as we know 3 teams get relegated in EPL each year so 6 teams should have less amount of matches.

matches["team"].value_counts()
```

```
Southampton                 72
Brighton and Hove Albion    72
Manchester United           72
West Ham United             72
Newcastle United            72
Burnley                     71
Leeds United                71
Crystal Palace              71
Manchester City             71
Wolverhampton Wanderers     71
Tottenham Hotspur           71
Arsenal                     71
Leicester City              70
Chelsea                     70
Aston Villa                 70
Everton                     70
Liverpool                   38
Fulham                      38
West Bromwich Albion        38
Sheffield United            38
Brentford                   34
Watford                     33
Norwich City                33
Name: team, dtype: int64
```

```
#Liverpool didn't get relegated in recent years so let's see why this team has less amount of matches

matches[matches["team"] == "Liverpool"]
```

	date	time	comp	round	day	venue	result	gf	ga	opponent	...	match report	notes	sh	sot	dist	fk	pk	pkatt	seas
1	2020-09-12	17:30	Premier League	Matchweek 1	Sat	Home	W	4.0	3.0	Leeds United	...	Match Report	NaN	20.0	4.0	17.0	0.0	2.0	2.0	20
2	2020-09-20	16:30	Premier League	Matchweek 2	Sun	Away	W	2.0	0.0	Chelsea	...	Match Report	NaN	17.0	5.0	17.7	1.0	0.0	0.0	20
4	2020-09-28	20:00	Premier League	Matchweek 3	Mon	Home	W	3.0	1.0	Arsenal	...	Match Report	NaN	21.0	9.0	16.8	0.0	0.0	0.0	20
6	2020-10-04	19:15	Premier League	Matchweek 4	Sun	Away	L	2.0	7.0	Aston Villa	...	Match Report	NaN	14.0	8.0	15.8	1.0	0.0	0.0	20
7	2020-10-17	12:30	Premier League	Matchweek 5	Sat	Away	D	2.0	2.0	Everton	...	Match Report	NaN	22.0	8.0	15.0	1.0	0.0	0.0	20
9	2020-10-24	20:00	Premier League	Matchweek 6	Sat	Home	W	2.0	1.0	Sheffield Utd	...	Match Report	NaN	17.0	5.0	18.2	1.0	0.0	0.0	20
11	2020-10-31	17:30	Premier League	Matchweek 7	Sat	Home	W	2.0	1.0	West Ham	...	Match Report	NaN	8.0	2.0	18.6	1.0	1.0	1.0	20
13	2020-11-08	16:30	Premier League	Matchweek 8	Sun	Away	D	1.0	1.0	Manchester City	...	Match Report	NaN	9.0	2.0	21.5	0.0	1.0	1.0	20
14	2020-11-22	19:15	Premier League	Matchweek 9	Sun	Home	W	3.0	0.0	Leicester City	...	Match Report	NaN	24.0	12.0	11.9	0.0	0.0	0.0	20
16	2020-11-28	12:30	Premier League	Matchweek 10	Sat	Away	D	1.0	1.0	Brighton	...	Match Report	NaN	6.0	2.0	20.9	0.0	0.0	0.0	20
18	2020-12-06	19:15	Premier League	Matchweek 11	Sun	Home	W	4.0	0.0	Wolves	...	Match Report	NaN	11.0	6.0	16.6	1.0	0.0	0.0	20
20	2020-12-13	16:30	Premier League	Matchweek 12	Sun	Away	D	1.0	1.0	Fulham	...	Match Report	NaN	11.0	5.0	20.0	1.0	1.0	1.0	20
21	2020-12-16	20:00	Premier League	Matchweek 13	Wed	Home	W	2.0	1.0	Tottenham	...	Match Report	NaN	17.0	11.0	15.5	0.0	0.0	0.0	20
22	2020-12-19	12:30	Premier League	Matchweek 14	Sat	Away	W	7.0	0.0	Crystal Palace	...	Match Report	NaN	14.0	7.0	13.2	1.0	0.0	0.0	20
23	2020-12-27	16:30	Premier League	Matchweek 15	Sun	Home	D	1.0	1.0	West Brom	...	Match Report	NaN	17.0	2.0	17.8	2.0	0.0	0.0	20
24	2020-12-30	20:00	Premier League	Matchweek 16	Wed	Away	D	0.0	0.0	Newcastle Utd	...	Match Report	NaN	11.0	4.0	16.7	0.0	0.0	0.0	20
25	2021-01-04	20:00	Premier League	Matchweek 17	Mon	Away	L	0.0	1.0	Southampton	...	Match Report	NaN	17.0	1.0	14.3	0.0	0.0	0.0	20
	2021		Premier	Matchweek						Manchester		Match								

```
#so we can see that we are missing data of this season (22-23).
matches["round"].value_counts()
```

```
Matchweek 1     39
Matchweek 16    39
Matchweek 34    39
Matchweek 32    39
Matchweek 31    39
Matchweek 29    39
Matchweek 28    39
Matchweek 26    39
Matchweek 25    39
Matchweek 24    39
Matchweek 23    39
Matchweek 2     39
Matchweek 19    39
Matchweek 17    39
Matchweek 20    39
Matchweek 15    39
Matchweek 5     39
Matchweek 3     39
Matchweek 13    39
Matchweek 12    39
Matchweek 4     39
Matchweek 11    39
Matchweek 10    39
Matchweek 9     39
Matchweek 8     39
Matchweek 14    39
Matchweek 7     39
Matchweek 6     39
Matchweek 30    37
Matchweek 27    37
Matchweek 22    37
Matchweek 21    37
Matchweek 18    37
Matchweek 33    32
Matchweek 35    20
Matchweek 36    20
Matchweek 37    20
Matchweek 38    20
Name: round, dtype: int64
```

```
#Now we know where our missing rows went, we lack some rows of some matchweeks
```

```
#CLEANING DATA FOR MACHING LEARNING
matches.dtypes
```

```
date             object
time             object
comp             object
round            object
day              object
venue            object
result           object
gf              float64
ga              float64
opponent         object
xg              float64
xga             float64
poss            float64
attendance      float64
captain          object
formation        object
referee          object
match report     object
notes           float64
sh              float64
sot             float64
dist            float64
fk              float64
pk              float64
pkatt           float64
season            int64
team             object
dtype: object
```

```
#ML algorithms can't work with objects. So we need to convert them to workable functions
matches["date"] = pd.to_datetime(matches["date"]) # not creating new column but overwritting the existing one
matches.dtypes
```

```
date            datetime64[ns]
time                    object
comp                    object
round                   object
day                     object
venue                   object
result                  object
gf                     float64
ga                     float64
opponent                object
xg                     float64
xga                    float64
poss                   float64
attendance             float64
captain                 object
formation               object
referee                 object
match report            object
notes                  float64
sh                     float64
sot                    float64
dist                   float64
fk                     float64
pk                     float64
pkatt                  float64
season                   int64
team                    object
dtype: object
```

```
#CREATING PREDICTORS FOR ML
matches["venue_code"] = matches["venue"].astype("category").cat.codes #converting strings into categories and converting categories into numb
```

```
matches
```

	date	time	comp	round	day	venue	result	gf	ga	opponent	...
1	2021-08-15	16:30	Premier League	Matchweek 1	Sun	Away	L	0.0	1.0	Tottenham	...
2	2021-08-21	15:00	Premier League	Matchweek 2	Sat	Home	W	5.0	0.0	Norwich City	...
3	2021-08-28	12:30	Premier League	Matchweek 3	Sat	Home	W	5.0	0.0	Arsenal	...
4	2021-09-11	15:00	Premier League	Matchweek 4	Sat	Away	W	1.0	0.0	Leicester City	...
6	2021-09-18	15:00	Premier League	Matchweek 5	Sat	Home	D	0.0	0.0	Southampton	...
...	...	...	...	...	...	...	...	...	...	...	...
38	2021-05-02	19:15	Premier League	Matchweek 34	Sun	Away	L	0.0	4.0	Tottenham	...
39	2021-05-08	15:00	Premier League	Matchweek 35	Sat	Home	L	0.0	2.0	Crystal Palace	...

```
matches["opp_code"] = matches["opponent"].astype("category").cat.codes
matches
```

	date	time	comp	round	day	venue	result	gf	ga	opponent	...
1	2021-08-15	16:30	Premier League	Matchweek 1	Sun	Away	L	0.0	1.0	Tottenham	...
2	2021-08-21	15:00	Premier League	Matchweek 2	Sat	Home	W	5.0	0.0	Norwich City	...
3	2021-08-28	12:30	Premier League	Matchweek 3	Sat	Home	W	5.0	0.0	Arsenal	...
4	2021-09-11	15:00	Premier League	Matchweek 4	Sat	Away	W	1.0	0.0	Leicester City	...
6	2021-09-18	15:00	Premier League	Matchweek 5	Sat	Home	D	0.0	0.0	Southampton	...
...	...	...	...	...	...	...	...	...	...	...	...
38	2021-05-02	19:15	Premier League	Matchweek 34	Sun	Away	L	0.0	4.0	Tottenham	...
39	2021-05-08	15:00	Premier League	Matchweek 35	Sat	Home	L	0.0	2.0	Crystal Palace	...

```
matches["hour"] = matches["time"].str.replace(":.+", "", regex=True).astype("int") #replace the colon and minutes(just keep the hour) with no
matches["day_code"] = matches ["date"].dt.dayofweek #changing weekdays with numbers
matches
```

	date	time	comp	round	day	venue	result	gf	ga	opponent	...
1	2021-08-15	16:30	Premier League	Matchweek 1	Sun	Away	L	0.0	1.0	Tottenham	...
2	2021-08-21	15:00	Premier League	Matchweek 2	Sat	Home	W	5.0	0.0	Norwich City	...
3	2021-08-28	12:30	Premier League	Matchweek 3	Sat	Home	W	5.0	0.0	Arsenal	...
4	2021-09-11	15:00	Premier League	Matchweek 4	Sat	Away	W	1.0	0.0	Leicester City	...
6	2021-09-18	15:00	Premier League	Matchweek 5	Sat	Home	D	0.0	0.0	Southampton	...
...	...	...	...	...	...	...	...	...	...	...	...
38	2021-05-02	19:15	Premier League	Matchweek 34	Sun	Away	L	0.0	4.0	Tottenham	...
39	2021-05-08	15:00	Premier League	Matchweek 35	Sat	Home	L	0.0	2.0	Crystal Palace	...

```
matches["target"] = (matches["result"] == "W").astype("int") #Set up a target which we are going to predict (e.g. our team won or not)
#win as 1 and draw or loss as 0
matches
```

	date	time	comp	round	day	venue	result	gf	ga	opponent	...
1	2021-08-15	16:30	Premier League	Matchweek 1	Sun	Away	L	0.0	1.0	Tottenham	...
2	2021-08-21	15:00	Premier League	Matchweek 2	Sat	Home	W	5.0	0.0	Norwich City	...
3	2021-08-28	12:30	Premier League	Matchweek 3	Sat	Home	W	5.0	0.0	Arsenal	...
4	2021-09-11	15:00	Premier League	Matchweek 4	Sat	Away	W	1.0	0.0	Leicester City	...
6	2021-09-18	15:00	Premier League	Matchweek 5	Sat	Home	D	0.0	0.0	Southampton	...
...	...	...	...	...	...	...	...	...	...	...	...
38	2021-05-02	19:15	Premier League	Matchweek 34	Sun	Away	L	0.0	4.0	Tottenham	...
39	2021-05-08	15:00	Premier League	Matchweek 35	Sat	Home	L	0.0	2.0	Crystal Palace	...

```
#CREATING INITIAL ML MODEL
from sklearn.ensemble import RandomForestClassifier #RandomForest can pick up Non-Linear type of data
rf = RandomForestClassifier(n_estimators=90, min_samples_split=10, random_state=1) #RF-> Series of Decision Trees but each DT has slightly di
#min_samples_split --> number of samples we want to have in a leaf of the DT before spliting the node, the higher this is the less likely to
#RF has lot of random parameters in it. If we set a random  satate it means if we run the RF multiple times we would get the same result.
train = matches[matches["date"] < '2022-01-01'] #time series data. Make sure All the data in the test set comes after the training set.
test = matches[matches["date"] > '2022-01-01']
#Why split up into train and test?
# We want the algorithm to do well in the predicting future matches which is why we test it out on the data that it hasn't been trained on. T
predictors = ["venue_code", "opp_code", "hour", "day_code"] #list of the predictor columns we have created
rf.fit(train[predictors], train["target"]) #fit our RF model. (the .fit method going to train our RF model with predictors)
RandomForestClassifier(min_samples_split=10, n_estimators=50, random_state=1)
preds = rf.predict(test[predictors])
#check accuracy in percentage
from sklearn.metrics import accuracy_score #what percentage of the time was your prediction accurate
acc = accuracy_score(test["target"],preds)
acc
```

```
#see in which situation our accuracy was high or low, need to create a Data frame for that
combined = pd.DataFrame(dict(actual=test["target"],prediction=preds))
pd.crosstab(index=combined["actual"],columns=combined["prediction"]) #we can see we were right most of the time about loss or draw but less r
```

prediction	0	1
actual		
0	141	31
1	76	28

```
#Revise our accuracy metric
from sklearn.metrics import precision_score #tells us when we predicted a win, what percentage of time it wins
precision_score(test["target"], preds) #47% is not good, not great precision
```

```
0.4745762711864407
```

```
#IMPROVING PRECISION WITH ROLLING AVERAGES --> using more predictors
#split the matches dataframe up by team. compute the rolling avg. (how many shots per goal, how many threats etc.)
grouped_matches = matches.groupby("team") #creates one dataframe for 1 squad in data.
group = grouped_matches.get_group("Manchester City")
group
```

	date	time	comp	round	day	venue	result	gf	ga	opponent	...
1	2021-08-15	16:30	Premier League	Matchweek 1	Sun	Away	L	0.0	1.0	Tottenham	...
2	2021-08-21	15:00	Premier League	Matchweek 2	Sat	Home	W	5.0	0.0	Norwich City	...
3	2021-08-28	12:30	Premier League	Matchweek 3	Sat	Home	W	5.0	0.0	Arsenal	...
4	2021-09-11	15:00	Premier League	Matchweek 4	Sat	Away	W	1.0	0.0	Leicester City	...
6	2021-09-18	15:00	Premier League	Matchweek 5	Sat	Home	D	0.0	0.0	Southampton	...
...	...	...	...	...	...	...	...	...	...	...	...
	2021-		Premier	Matchweek						Crystal	

```
#if we were in matchweek 4 how did City do in previous 3 matches and use that information to feed the algorithm
def rolling_averages(group, cols, new_cols): #new_cols for assigning the rolling avgs to.
  group = group.sort_values("date")
  rolling_stats = group[cols].rolling(3, closed='left').mean() #leaves the row that is going to predicted
  group[new_cols] = rolling_stats
  group = group.dropna(subset=new_cols) #drops missing values
  return group
cols = ["gf", "ga", "sh", "sot", "dist", "fk", "pk", "pkatt"] #cols that is going to be computed rolling avgs for
new_cols = [f"{c}_rolling" for c in cols]
new_cols #going to be created rolling avgs in them
```

```
['gf_rolling',
 'ga_rolling',
 'sh_rolling',
 'sot_rolling',
 'dist_rolling',
 'fk_rolling',
 'pk_rolling',
 'pkatt_rolling']
```

```
rolling_averages(group, cols, new_cols) #Only for Manchester City
```

	date	time	comp	round	day	venue	result	gf	ga	opponent	...	d
5	2020-10-17	17:30	Premier League	Matchweek 5	Sat	Home	W	1.0	0.0	Arsenal	...	
7	2020-10-24	12:30	Premier League	Matchweek 6	Sat	Away	D	1.0	1.0	West Ham	...	
9	2020-10-31	12:30	Premier League	Matchweek 7	Sat	Away	W	1.0	0.0	Sheffield Utd	...	
11	2020-11-08	16:30	Premier League	Matchweek 8	Sun	Home	D	1.0	1.0	Liverpool	...	
12	2020-11-21	17:30	Premier League	Matchweek 9	Sat	Away	L	0.0	2.0	Tottenham	...	
...	...	...	...	...	...	...	...	...	...	...	...	
42	2022-03-14	20:00	Premier League	Matchweek 29	Mon	Away	D	0.0	0.0	Crystal Palace	...	
44	2022-04-02	15:00	Premier League	Matchweek 31	Sat	Away	W	2.0	0.0	Burnley	...	

```
#apply this to all of our teams
matches_rolling = matches.groupby("team").apply(lambda x: rolling_averages(x, cols, new_cols))
matches_rolling
```

		date	time	comp	round	day	venue	result	gf	ga	
team											
Arsenal	**6**	2020-10-04	14:00	Premier League	Matchweek 4	Sun	Home	W	2.0	1.0	
	7	2020-10-17	17:30	Premier League	Matchweek 5	Sat	Away	L	0.0	1.0	M
	9	2020-10-25	19:15	Premier League	Matchweek 6	Sun	Home	L	0.0	1.0	
	11	2020-11-01	16:30	Premier League	Matchweek 7	Sun	Away	W	1.0	0.0	M
	13	2020-11-08	19:15	Premier League	Matchweek 8	Sun	Home	L	0.0	3.0	.
...	**...**	...	...	...	...	...	...	...	...	...	
Wolverhampton Wanderers	**32**	2022-03-13	14:00	Premier League	Matchweek 29	Sun	Away	W	1.0	0.0	
	33	2022-03-18	20:00	Premier League	Matchweek 30	Fri	Home	L	2.0	3.0	

```
#matches_rolling = matches_rolling.droplevel('team') --> makes it difficult to work with name, so we drop this extra index levels
# we want unique values in our index
matches_rolling.index = range(matches_rolling.shape[0])
matches_rolling
```

	date	time	comp	round	day	venue	result	gf	ga	opponent	...
0	2020-10-04	14:00	Premier League	Matchweek 4	Sun	Home	W	2.0	1.0	Sheffield Utd	...
1	2020-10-17	17:30	Premier League	Matchweek 5	Sat	Away	L	0.0	1.0	Manchester City	...
2	2020-10-25	19:15	Premier League	Matchweek 6	Sun	Home	L	0.0	1.0	Leicester City	...
3	2020-11-01	16:30	Premier League	Matchweek 7	Sun	Away	W	1.0	0.0	Manchester Utd	...
4	2020-11-08	19:15	Premier League	Matchweek 8	Sun	Home	L	0.0	3.0	Aston Villa	...
...	...	...	...	...	...	...	...	...	...	...	...
1312	2022-03-13	14:00	Premier League	Matchweek 29	Sun	Away	W	1.0	0.0	Everton	...
1313	2022-03-18	20:00	Premier League	Matchweek 30	Fri	Home	L	2.0	3.0	Leeds United	...

```
#RETRAINING OUR ML MODEL
def make_predictions(data, predictors):
  train= data[data["date"] < '2022-01-01']
  test = data[data["date"] > '2022-01-01']
  rf.fit(train[predictors], train["target"])
  preds = rf.predict(test[predictors])
  combined = pd.DataFrame(dict(actual=test["target"], predicted=preds), index=test.index)
  precision = precision_score(test["target"], preds)
  return combined, precision
combined, precision = make_predictions(matches_rolling, predictors + new_cols)
precision
```

```
    0.625
```

```
combined #can't really see if we are mispredicting any teams results particularly
```

	actual	predicted
55	0	0
56	1	0
57	1	0
58	1	1
59	1	1
...	...	...
1312	1	0

```
# So we are merging team, date , opponent and result based on index
combined = combined.merge(matches_rolling[["date", "team", "opponent", "result"]],left_index=True, right_index=True)
combined
```

	actual	predicted	date	team	opponent	result
55	0	0	2022-01-23	Arsenal	Burnley	D
56	1	0	2022-02-10	Arsenal	Wolves	W
57	1	0	2022-02-19	Arsenal	Brentford	W
58	1	1	2022-02-24	Arsenal	Wolves	W
59	1	1	2022-03-06	Arsenal	Watford	W
...	...	...	...	...	...	...
1312	1	0	2022-03-13	Wolverhampton Wanderers	Everton	W
1313	0	0	2022-03-18	Wolverhampton Wanderers	Leeds United	L
1314	1	0	2022-04-02	Wolverhampton Wanderers	Aston Villa	W
1315	0	0	2022-04-08	Wolverhampton Wanderers	Newcastle Utd	L
1316	0	0	2022-04-24	Wolverhampton Wanderers	Burnley	L

276 rows × 6 columns

```
#COMBINING HOME AND AWAY PREDICTIONS
#before combining them we have to normalize the name columns
class MissingDict(dict):   # Pandas Math dictionary by default will remove missing name and all its data but we would create a mapping dictio
  __missing__ = lambda self, key: key
map_values = {
    "Brighton and Hove Albion" : "Brighton",
    "Manchester United": "Manchester Utd",
    "Newcastle United": "Newcastle Utd",
    "West Ham United": "West Ham",
    "Wolverhampton Wanderers": "Wolves"
}
mapping = MissingDict(**map_values)
mapping["Wolverhampton Wanderers"]
```

```
'Wolves'
```

```
combined["new_team"] = combined["team"].map(mapping)
combined
```

	actual	predicted	date	team	opponent	result	new_team
55	0	0	2022-01-23	Arsenal	Burnley	D	Arsenal
			2022-				

```
merged = combined.merge(combined, left_on=["date", "new_team"], right_on=["date", "opponent"]) #look for new team field and merge that with t
merged
```

	actual_x	predicted_x	date	team_x	opponent_x	result_x	new_team_x
0	0	0	2022-01-23	Arsenal	Burnley	D	Arsenal
1	1	0	2022-02-10	Arsenal	Wolves	W	Arsenal
2	1	0	2022-02-19	Arsenal	Brentford	W	Arsenal
3	1	1	2022-02-24	Arsenal	Wolves	W	Arsenal
4	1	1	2022-03-06	Arsenal	Watford	W	Arsenal
...	...	...	...	...	...	...	...
242	1	0	2022-03-13	Wolverhampton Wanderers	Everton	W	Wolves
243	0	0	2022-03-18	Wolverhampton Wanderers	Leeds United	L	Wolves

```
merged[(merged["predicted_x"] == 1) & (merged["predicted_y"] == 0)]["actual_x"].value_counts() # showing only those results where algorithm h
```

```
1    24
0    12
Name: actual_x, dtype: int64
```

```
24/36 #Precision Rate
```

```
0.6666666666666666
```

Colab paid products - Cancel contracts here