```
###PACKAGES#
##
                library(pacman)
                p_load(rvest,tidyverse,janitor,tidymodels,broom,
                       rpart,rpart.plot)

                final_df$allstar= as.factor(final_df$allstar)
                ###DATA###
                ##Read in data, use janitor::clean_names for renaming columns

                #background info
                playerstats_df <- read.csv("data/player_data.csv") %>%
                clean_names()%>%rename(player = name)

                #physical characteristics
                players_df <- read.csv("data/Players.csv") %>% clean_names() %>%
                  mutate(player = gsub(" +$","",gsub("[^[:alpha:]]"," ",player)))

                #season stats
                seasonstats_df <- read.csv("data/Seasons_Stats.csv") %>%
                  #Change year format to be more specific (from xxxx-yy to xxxx-yyyy)
                  mutate(season = paste(Year-1,Year,sep = "-")) %>%
                  #reorder columns for convenience
                  .[,c(1,2,54,3:53)] %>% clean_names() %>%
                  #Only considering data after the 1999 season
                  filter(year>1999) %>%

                  #Clean up player names
                  mutate(player = gsub(" +$","",gsub("[^[:alpha:]]"," ",player)))

                #Read in wikipedia scrapped data
                #MVP data
                mvp <-
                read_html("https://en.wikipedia.org/wiki/NBA_Most_Valuable_Player_Award")
                %>%
                  html_nodes(xpath = "//*[@id='mw-content-text']/div[1]/table[4]") %>%
                  html_table() %>% data.frame() %>% clean_names() %>%
                  #clean up season and player variables
                  mutate(season = gsub("[^[:alnum:]]","-19",season),
                         player = gsub(" +$","",gsub("[^[:alpha:]]"," ",player))) %>%
                  #rename player variable to ensure smooth joining with other dfs
                  rename("year_mvp" = "player") %>%
                  .[,c(1,2)]
```

```
mvp$season[45:65] <- gsub("^([0-9]{4})([^[:alnum:]]{1}[0-9]{2})([0-
9]{2})$","\\1-20\\3",mvp$season[45:65])
mvp$year_mvp[44] <- "Karl Malone"
mvp$year_mvp[56] <- "Derrick Rose"

#All-star selections
allstars <- read_html("https://en.wikipedia.org/wiki/List_of_NBA_All-
Stars") %>%
  html_nodes(xpath = "//*[@id='mw-content-text']/div[1]/table[2]") %>%
  html_table() %>% data.frame() %>% clean_names() %>%
  #clean up player variable, and prepare allstar selections for future use
  mutate(player = gsub(" +$","",gsub("[^[:alpha:]]"," ",player)),
         selections = strsplit(selections_c,split = ";"),
         selections_c = NULL) %>%
  .[,c(1,5)]
#manually fix minor issues from webscraping
allstars$player[1] <- "Kareem Abdul Jabbar"
allstars$player[19] <- "Hakeem Olajuwon"
allstars$player[430] <- "Metta World Peace"

#merging the player df's
df_halffull <- left_join(seasonstats_df, players_df,playerstats_df, by =
"player")

#create mvp column with TRUE/FALSE values
full_df <- left_join(df_halffull, mvp, by ="season") %>%
  mutate(mvp = (year_mvp == player))

#Create allstar column with TRUE/FALSE values
full_df$allstar <- FALSE
for (i in 1:nrow(allstars)) {

  #Pickup selections in their source format from webscrapped table
  selections <- allstars$selections[[i]]

  #Turn the format into yyyy:yyyy for further use
  selections_seq <- gsub("[^[:alnum:]]",":",gsub(" ","",selections))

  #put years selected into a useable format: a df
  #player= player selected; selections = years selected for allstar game
  years_selected <- data.frame(
    player = allstars$player[i],
    selections = unlist(lapply(selections_seq,function(x) {eval(parse(text
= x))}))
  )

  #turn values of full_df's allstar column to true based on membership in
years_selected
  full_df[(full_df$player == years_selected$player[1] &
             full_df$year %in% years_selected$selections),"allstar"] <-
TRUE
}

#create final df with finalized variables
final_df <- full_df %>%
  #create an age column
  mutate(age = year-born) %>%
  #reorder columns for convenience
  .[,c(2:4,63:64,56:59,60,61,5:54)] %>%
  select(-c(blanl,blank2)) %>%
  mutate(mvp = as_factor(as.numeric(mvp)), allstar =
as_factor(as.numeric(allstar)))

#for loop creating dummy variables for different positions
for (i in c("C","PG","PF","SG","SF")) {
  eval(parse(text = paste("final_df$",i,"<-
as.numeric(grepl('",i,"',final_df$pos))",sep = "")))
}

###Elasticnet Logistic Regression###
set.seed(8237)

#split df into training and testing sets by randomly choosing years, each
as their own sample
#18 years possible, we will be using 80% (rounded up to 15 total years) for
our training and the remainder for testing.
train_years <- sample(unique(final_df$year),15) %>%
  sort()
test_years <- unique(final_df$year)[!unique(final_df$year) %in%
train_years] %>%
  sort()

final_train <- final_df[final_df$year %in% train_years,]
final_test <- final_df[final_df$year %in% test_years,]
```

```r
#Use each year as it's own sample
final_cv <- final_train %>% group_vfold_cv(group = year)

#Create recipe and prepare it for use
final_recipe <- final_train %>% recipe(allstar ~ .) %>%
  #Remove problematic variables
  step_rm(player) %>% step_rm(collage) %>% step_rm(pos) %>%
  step_rm(birth_city) %>% step_rm(birth_state) %>%
  step_rm(season) %>% step_rm(tm) %>%
  #normalize numeric variables
  step_normalize(all_predictors() & all_numeric()) %>%
  #turn categorical variables into dummy variables
  step_dummy(all_predictors() & all_nominal()) %>%
  #impute categorical variables
  step_modeimpute(all_predictors()&all_nominal()) %>%
  #impute numeric variables
  step_meanimpute(all_predictors()&all_numeric())
final_clean <- final_recipe %>% prep() %>% juice()

#prepare elasticnet logit model
model_en <- logistic_reg(penalty = tune(), mixture = tune()) %>%
  set_engine("glmnet")

#prepare elasticnet logit workflow
workflow_en = workflow() %>%
  add_model(model_en) %>%
  add_recipe(final_recipe)

#calculate the best models
cv_en = workflow_en %>%
  tune_grid(
    final_cv,
    grid = grid_regular(mixture(), penalty(), levels = 3),
    metrics = metric_set(accuracy)
  )

final_en = workflow_en %>%
  finalize_workflow(select_best(cv_en, 'accuracy'))

#Fitting the final model
final_fit_en = final_en %>% fit(data = final_train)

#Predict onto the test data
```

```
y_hat = final_fit_en %>% predict(new_data = final_test, type ="class")

cm_logistic = conf_mat(
  data = tibble(
    y_hat = y_hat %>% unlist(),
    y = final_test$allstar
  ),
  truth = y, estimate = y_hat
)

#view confusion matrix
cm_logistic


#confusion matrix with probabilities

#Predict onto the test data
p_hatlogit = final_fit_en %>% predict(new_data = final_test, type ="prob")

cm_logisticp = conf_mat(
  data = tibble(
    p_hatlogit = p_hatlogit$.pred_2,
    plogit = final_test$allstar
  ),
  truth = plogit, estimate = p_hatlogit
)


#create logit dataframe
logit_df = data.frame(
  player = final_test$player,
  year = final_test$year,
  allstar = p_hatlogit$.pred_2
)


##graph 2006 predictions
logit_df%>% filter(allstar>.9, year==2006)%>%ggplot(aes(x=allstar, y=
reorder(player,(allstar)),fill=player))+geom_col()+theme_classic(base_size
= 12)+scale_fill_viridis_d() +xlab("Prediction") +ylab("Player")+
ggtitle("2006 Allstar Predictions Logistic Regression Model")

#logistic visual
```

```
cm_logistic$table %>% as.data.frame() %>%
  mutate(result = c("True Negative","False Positive","False Negative","True
Positive")) %>%
  ggplot(aes(x = result, y = Freq)) +
  geom_bar(stat = "identity") +xlab("Result")+ylab("Frequency")
+ggtitle("Allstar Predictions from Logistic
Model")+theme_bw()+scale_fill_brewer()

#graph without true negatives
cm_logistic$table %>% as.data.frame() %>%
  mutate(result = c("True Negative","False Positive","False Negative","True
Positive")) %>%
  filter(result!= "True Negative")%>%
  ggplot(aes(x = result, y = Freq, fill =result)) +
  geom_bar(stat = "identity") + xlab("Result")+ylab("Frequency")
+ggtitle("Allstar Predictions from Logistic
Model")+theme_bw()+scale_fill_brewer()

#logistic
cm_logistic$table %>% as.data.frame() %>%
  mutate(result = c("True Negative","False Positive","False Negative","True
Positive")) %>%
  filter(result!= "True Negative")%>%
  ggplot(aes(x = result, y = Freq, fill =result)) + labs(fill = "Result")+
  geom_bar(stat = "identity") + xlab("Result")+ylab("Frequency")
+ggtitle("Allstar Predictions from Logistic Model")+theme_classic(base_size
= 12)+scale_fill_manual(values=c( "#B54213","#161616", "#EF7F4D" ))


###Decision Tree###
#decision tree model
model_tree <- decision_tree(
  mode = "classification",
  cost_complexity = tune(),
  tree_depth = tune(),
  min_n = 10
) %>%
  set_engine("rpart")

#setup decision tree workflow
tree_workflow <- workflow() %>%
  add_model(model_tree) %>%
  add_recipe(final_recipe)
```

```r
#tune models
tree_cv_fit <- tree_workflow %>% tune_grid(
  final_cv,
  grid = expand_grid(
    cost_complexity = seq(0,.15,by = .05),
    tree_depth = c(1,5,10)
  ),
  metrics = metric_set(accuracy, roc_auc)
)

#select the best model based on accuracy
best_flow_tree <- tree_workflow %>%
  finalize_workflow(select_best(tree_cv_fit, metric = "accuracy")) %>%
  fit(data = final_train)

#pull the best decision tree model
best_tree <- best_flow_tree %>% pull_workflow_fit()

#plot decision tree
best_tree$fit %>% rpart.plot(cex = 1)
#Fitting the final model
final_fit_tree = best_flow_tree %>% fit(data = final_train)

#Predict onto the test data
y_hat_tree = final_fit_tree %>% predict(new_data = final_test, type
="class")

#confusion matrix for decision tree
cm_logistic_tree = conf_mat(
  data = tibble(
    y_hat = y_hat_tree %>% unlist(),
    y = final_test$allstar
  ),
  truth = y, estimate = y_hat
)

#decision tree visuals
cm_logistic_tree$table %>% as.data.frame() %>%
  mutate(result = c("True Negative","False Positive","False Negative","True
Positive")) %>%
  ggplot(aes(x = result, y = Freq)) +
  geom_bar(stat = "identity") +
  geom_text(aes(x = "False Negative",y = c(600,750,900,1050),label =
paste(result,"s: ",Freq,sep = "")))


cm_logistic_tree

#predict probabilities
p_hat_tree = final_fit_tree %>% predict(new_data = final_test, type
="prob")

#confusion matrix from estimates
cm_logistic_treep = conf_mat(
  data = tibble(
    p_hat = p_hat_tree %>% unlist(),
    p = final_test$allstar
  ),
  truth = p, estimate = p_hat
)


##create a decision tree data frame
tree_df = data.frame(
  player = final_test$player,
  year = final_test$year,
  allstar = p_hat_tree
)

##graph 06 predictions

tree_df%>% filter(allstar..pred_2>0.9,
year==2006)%>%ggplot(aes(x=allstar..pred_2, y=
reorder(player,(allstar..pred_2)),fill=player))+geom_col()+theme_classic(ba
se_size = 12)+scale_fill_viridis_d() +xlab("Prediction") +ylab("Player")+
ggtitle("2006 Allstar Predictions Decision Tree")


##graph without true negatives
cm_logistic_tree$table %>% as.data.frame() %>%
  mutate(result = c("True Negative","False Positive","False Negative","True
Positive")) %>%
  filter(result!= "True Negative")%>%
  ggplot(aes(x = result, y = Freq, fill =result)) +
```

```r
  geom_bar(stat = "identity") + xlab("Result")+ylab("Frequency")
+ggtitle("Allstar Predictions from Decision Tree
Model")+theme_bw()+scale_fill_brewer()

###Elasticnet Linear Regression###
#change allstar to numeric for compatibility with linear regression
final_df$allstar <- as.numeric(final_df$allstar)-1
final_train$allstar <- as.numeric(final_train$allstar)-1
final_test$allstar <- as.numeric(final_test$allstar)-1

#rerun cv splits for updated outcome class
final_cv2 <- final_train %>% group_vfold_cv(group = year)

#rerun recipe in case it needs to update for new class of outcome variable
final_recipe2 <- final_train %>% recipe(allstar ~ .) %>%
  #Remove problematic variables
  step_rm(player) %>% step_rm(collage) %>% step_rm(pos) %>%
  step_rm(birth_city) %>% step_rm(birth_state) %>%
  step_rm(season) %>% step_rm(tm) %>%
  #normalize numeric variables
  step_normalize(all_predictors() & all_numeric()) %>%
  #turn categorical variables into dummy variables
  step_dummy(all_predictors() & all_nominal()) %>%
  #impute categorical variables
  step_modeimpute(all_predictors()&all_nominal()) %>%
  #impute numeric variables
  step_meanimpute(all_predictors()&all_numeric())

#Elasticnet regression
model_en = linear_reg(penalty = tune(), mixture = tune()) %>%
set_engine("glmnet")

#Define the workflow
workflow_en = workflow() %>%
  add_model(model_en) %>%
  add_recipe(final_recipe2)



#run the model
cv_enlinreg = workflow_en %>%
  tune_grid(
    final_cv2,
    grid = grid_regular(mixture(), penalty(), levels = 5:5),
    metrics = metric_set(rmse)
  )

cv_en %>% collect_metrics() %>% arrange(mean)

#finalize
enlinreg_final <- workflow_enlinreg%>%
  finalize_workflow(select_best(cv_enlinreg, metric="rmse"))

#fit final model
enlinreg_final_fit = enlinreg_final%>%fit(data=final_train)

#predict onto test data
test_hat = enlinreg_final_fit %>% predict(new_data = final_test)

head(test_hat)


enlinreg_df = data.frame(
  player = final_test$player,
  year = final_test$year,
  allstar = test_hat
)

#view
enlinreg_df


##graph of 2006 predictions
enlinreg_df$player = with(enlinreg_df, reorder(player,.pred))
enlinreg_df%>% filter(.pred>1.65, year==2006)%>%ggplot(aes(x=.pred, y=
reorder(player,(.pred)),fill=player))+geom_col()+theme_classic(base_size =
12)+scale_fill_viridis_d() +xlab("Prediction") +ylab("Player")+
ggtitle("2006 Allstar Predictions Boosted Model")


final_test$allstar_reg <- test_hat$.pred


#Now, lets take the top 27 estimates for each year and guess those as our
allstars
```

```
for (i in unique(final_test$year)) {
  #generate cutoff value such that 25 players are above it
  cutoff <- final_test[final_test$year == i,] %>% select(allstar_reg) %>%
    pull() %>%
    sort(decreasing = TRUE) %>%
    .[25]

  #get vector of allstar regression results
  year_allstar_reg <- final_test[final_test$year == i,] %>%
    select(allstar_reg) %>% pull()

  #turn allstar_values into binary
  year_allstar_pred <- ifelse(year_allstar_reg>=cutoff,1,0)

  #if statement: add to allstar_pred_vec
  if (exists("allstar_pred_vec")) {
    allstar_pred_vec <- append(allstar_pred_vec,year_allstar_pred)
  }
  #if statement: create original allstar_pred_vec
  if (!exists("allstar_pred_vec")) {
    allstar_pred_vec <- year_allstar_pred
  }
}

#add predictions to df
final_test$allstar_pred <- allstar_pred_vec

#make predictions
y_hat_tree <- final_test$allstar_pred

#confusion matrix
cm_lin_reg = conf_mat(
  data = tibble(
    y_hat = as.factor(y_hat_tree) %>% unlist(),
    y = as.factor(final_test$allstar)
  ),
  truth = y, estimate = y_hat
)

#lin reg visuals
cm_lin_reg$table %>% as.data.frame() %>%
  mutate(result = c("True Negative","False Positive","False Negative","True
Positive")) %>%
  ggplot(aes(x = result, y = Freq)) +
  geom_bar(stat = "identity") +
  geom_text(aes(x = "False Negative",y = c(600,750,900,1050),label =
paste(result,"s: ",Freq,sep = "")))

cm_lin_reg$table %>% as.data.frame() %>%
  mutate(result = c("True Negative","False Positive","False Negative","True
Positive")) %>%
  filter(result!= "True Negative")%>%
  ggplot(aes(x = result, y = Freq, fill =result)) + labs(fill = "Result")+
  geom_bar(stat = "identity") + xlab("Result")+ylab("Frequency")
+ggtitle("Allstar Predictions from Elasticnet Linear Regression
Model")+theme_classic(base_size = 12)+scale_fill_manual(values=c(
"#B54213","#161616", "#EF7F4D" ))


### Boost model###
#define boost model
allstar_boost = boost_tree(
  mtry= NULL,
  trees=10,
  min_n =NULL,
  tree_depth = tune(),
  learn_rate = tune()
) %>% set_engine(
  engine = "xgboost") %>%
  set_mode("classification")
final_test$allstar <- as.factor(final_test$allstar)


#define workflow
allstar_boost_wf =
  workflow()%>% add_model(allstar_boost)%>%add_recipe(final_recipe)

#run model
cv_boost = allstar_boost_wf %>%
  tune_grid(
    final_cv,
    grid = grid_regular(tree_depth(), learn_rate(), levels = 5:5),
    metrics = metric_set(accuracy)
  )

#show the best model
```

```
cv_boost %>% show_best()

#finalize workflow and use it to predict onto test data
final_boost =
  allstar_boost_wf %>%
  finalize_workflow(select_best(cv_boost, "accuracy"))


#fit final model

final_fit_boost = final_boost %>% fit(data =final_train)


#predict onto test data
p_hat = final_fit_boost %>% predict(new_data = final_test, type="class")

#confusion matrix
cm_boost = conf_mat(
  data =tibble(
    p_hat = p_hat %>% unlist(),
    p = final_test$allstar
  ),
  truth = p, estimate = p_hat
)
#view matrix
cm_boost


head(p_hat)


##turn confusion matrix into a graph
cm_boost$table %>% as.data.frame() %>%
  mutate(result = c("True Negative","False Positive","False Negative","True
Positive")) %>%
  ggplot(aes(x = result, y = Freq, fill =result))
+theme_bw()+scale_fill_brewer() +geom_bar(stat = "identity") +
xlab("Result")+ylab("Frequency") +
  geom_text(aes(x = "False Negative",y = c(600,750,900,1050),label =
paste(result,"s: ",Freq,sep = "")))

##graph without true negatives
cm_boost$table %>% as.data.frame() %>%
  mutate(result = c("True Negative","False Positive","False Negative","True
Positive")) %>%
  filter(result!= "True Negative")%>%
  ggplot(aes(x = result, y = Freq, fill =result)) +
  geom_bar(stat = "identity") + xlab("Result")+ylab("Frequency")
+ggtitle("Allstar Predictions from Boosted
Model")+theme_bw()+scale_fill_brewer()
```