Predicting National Park Traffic

Jonathon Bowyer

Metis Project #2

Goal and Motivation

- Given historical data about park attendance, predict future attendance
- National Park Services (NPS) needs to plan for traffic
  - Inform decisions about when to close a park for maintenance
  - Plan how many employees are needed at any given point in the year
- Inform visitors how busy they can expect the park to be

Data and Data Manipulation

- Data on 51 National Parks
- Parks have per-month visitors since at least 1979 through April 2018
  - n = (39*12 + 4) * 51 = ~24,000

- Created variables representing:
  - LSTYRTOTAL = total yearly visitors at that park from the prior year
    - Baseline weight of how big/trafficked the park is
  - CHTYR = prior year's traffic growth rate at that park
    - Growth rate

Model

- Linear regression to estimate the traffic
  - The data appear to meet the Gauss-Markov assumptions
    - The features will have a linear relationship with the output
    - Data is the full set of what we're interested in, not a biased sampling
    - A large number of observations
    - The others we check after running the model, and appear to be satisfied

```
y, X = patsy.dmatrices('VISITORS ~ LSTYRTOTAL + CHTYR + MONTH',
                       data=np_df, return_type="dataframe")
```

Model Output

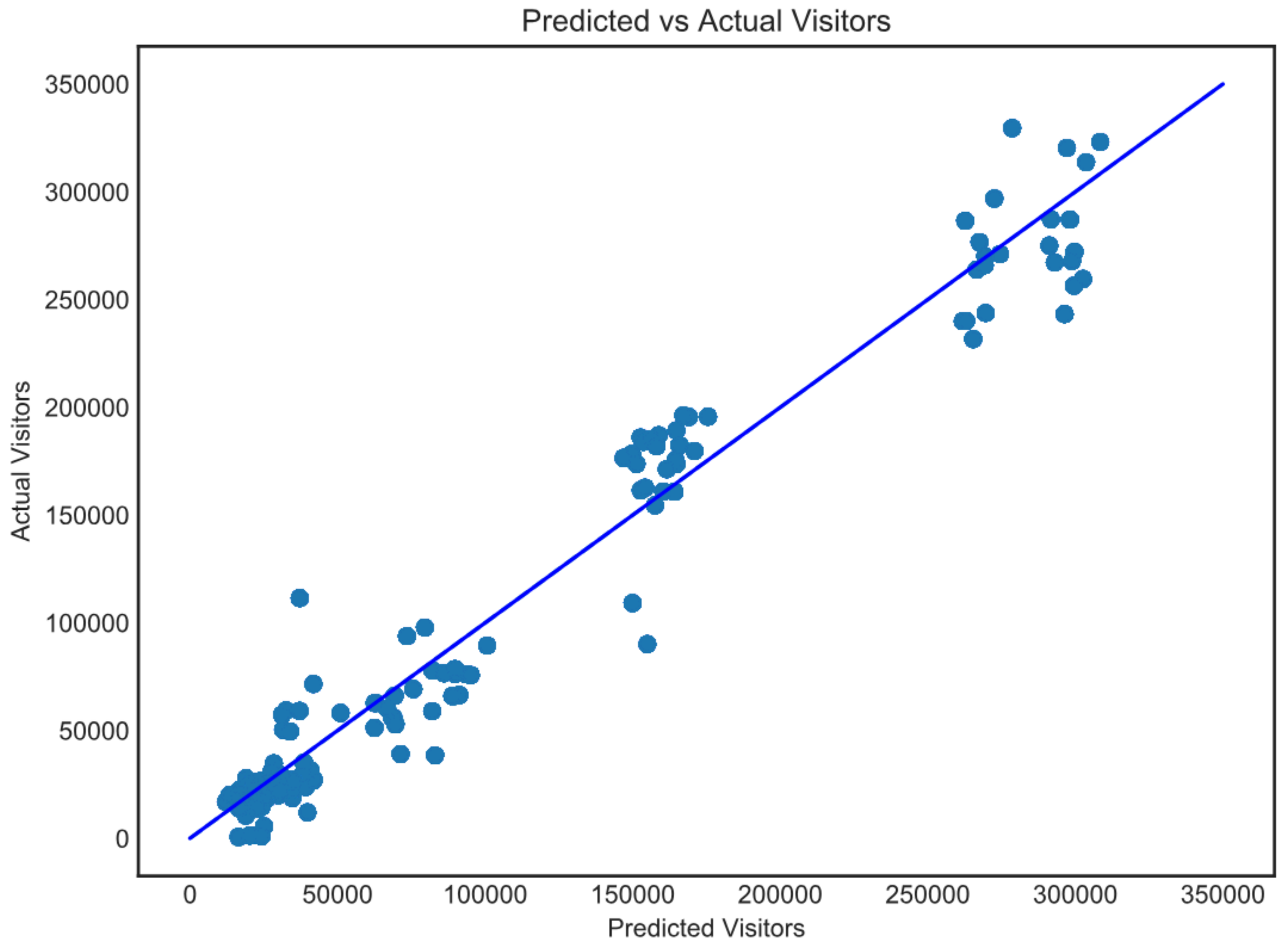
Dep. Variable:	VISITORS	R-squared:	0.952
Model:	OLS	Adj. R-squared:	0.952
Method:	Least Squares	F-statistic:	3.837e+04
Date:	Fri, 20 Jul 2018	Prob (F-statistic):	0.00
Time:	09:47:04	Log-Likelihood:	2472.5
No. Observations:	25154	AIC:	-4917.
Df Residuals:	25140	BIC:	-4803.
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.7836	0.005	-161.798	0.000	-0.793	-0.774
LSTYRTOTAL	0.0990	0.002	56.300	0.000	0.096	0.102
CHTYR	0.0020	0.002	0.798	0.425	-0.003	0.007
FEB	-0.0417	0.007	-6.095	0.000	-0.055	-0.028
MAR	-0.0064	0.007	-0.937	0.349	-0.020	0.007
APR	0.0492	0.007	7.224	0.000	0.036	0.063
MAY	0.4563	0.007	67.042	0.000	0.443	0.470
JUN	1.3765	0.007	202.247	0.000	1.363	1.390
JUL	2.5123	0.007	369.134	0.000	2.499	2.526
AUG	2.7613	0.007	405.714	0.000	2.748	2.775
SEP	1.5024	0.007	220.740	0.000	1.489	1.516
OCT	0.6153	0.007	90.404	0.000	0.602	0.629
NOV	0.1931	0.007	28.370	0.000	0.180	0.206
DEC	-0.0088	0.007	-1.298	0.194	-0.022	0.005

Omnibus:	7478.317	Durbin-Watson:	1.522
Prob(Omnibus):	0.000	Jarque-Bera (JB):	55021.738
Skew:	1.232	Prob(JB):	0.00
Kurtosis:	9.814	Cond. No.	13.1

- P statistics look quite good, could perhaps drop growth indicator
- Coefficients for Yearly Total and Change in Year Total have the right sign
- Cross Validation
  - Time series data, split data before and after a certain year
  - Across multiple CVs, Adj R^2 for predictions ranged from .95 to .97
  - Higher MSE for train than test, not overfitting!

Model Validation



Conclusions and Recommendations

- Visitation can be predicted relatively accurately
- Coefficients for months, especially sans normalization, are useful to interpret for trends

FEB	-4077.9494
MAR	-623.2494
APR	4803.7103
MAY	4.458e+04
JUN	1.345e+05
JUL	2.455e+05
AUG	2.698e+05
SEP	1.468e+05
OCT	6.012e+04
NOV	1.886e+04
DEC	-863.5118

- Generally, we want to do maintenance and have low staff in January, February, March or December
- Need the most staff in the summer months (matches intuition)

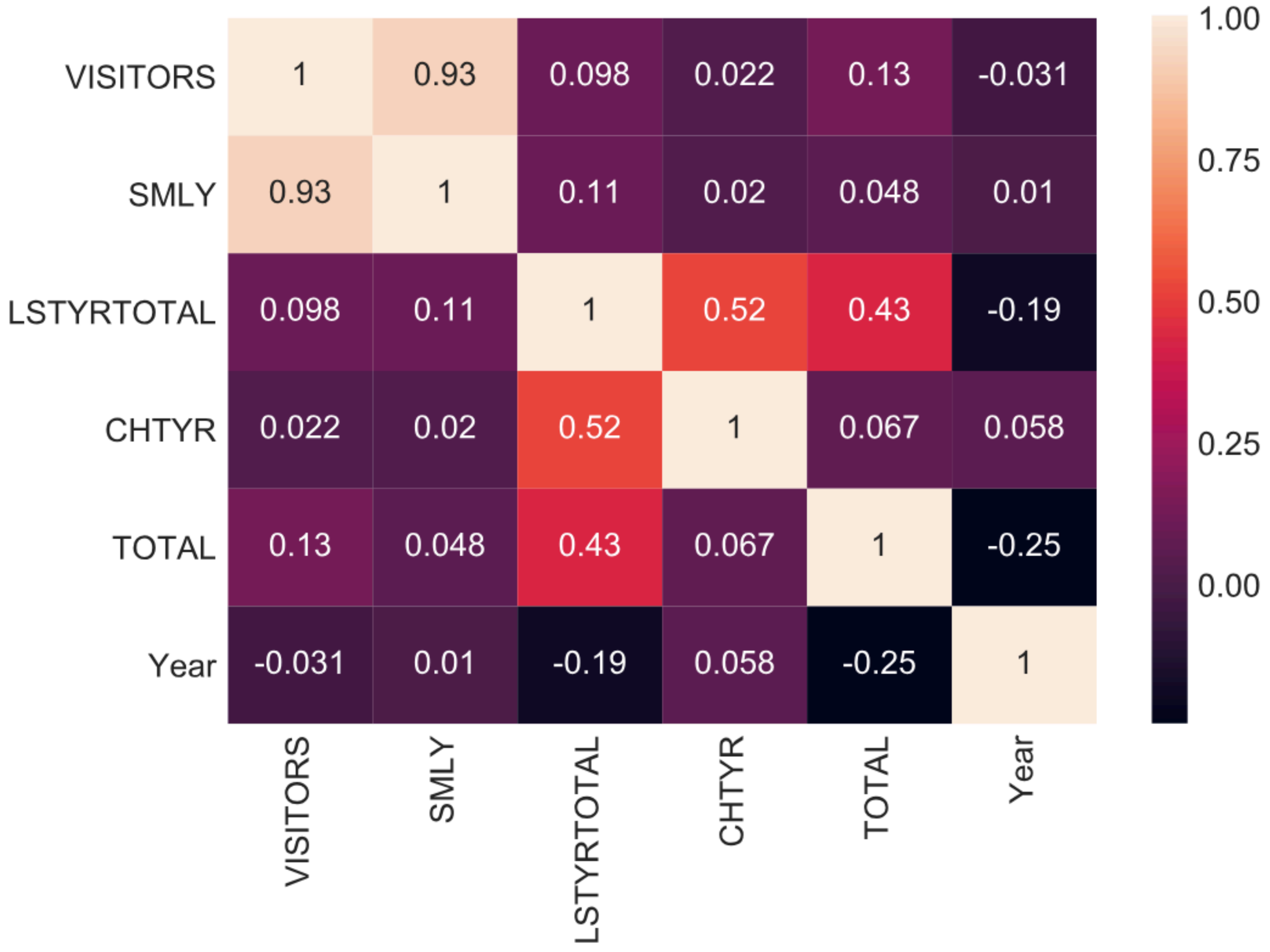
Future Work

- If desired, work is highly iterable for various other park sites (National Historic Sites, National Monuments, and so forth)
  - Could help with similar problems in these sites
    - Planning days to do maintenance
    - Employee staffing
    - Consumer desire to plan around traffic
- Incorporate more features, especially weather data
  - Suggested variables: monthly highs and lows, and days with precipitation
  - Beware multicollinearity with Month variable
- Investigate middle clustering of predictions

Appendix

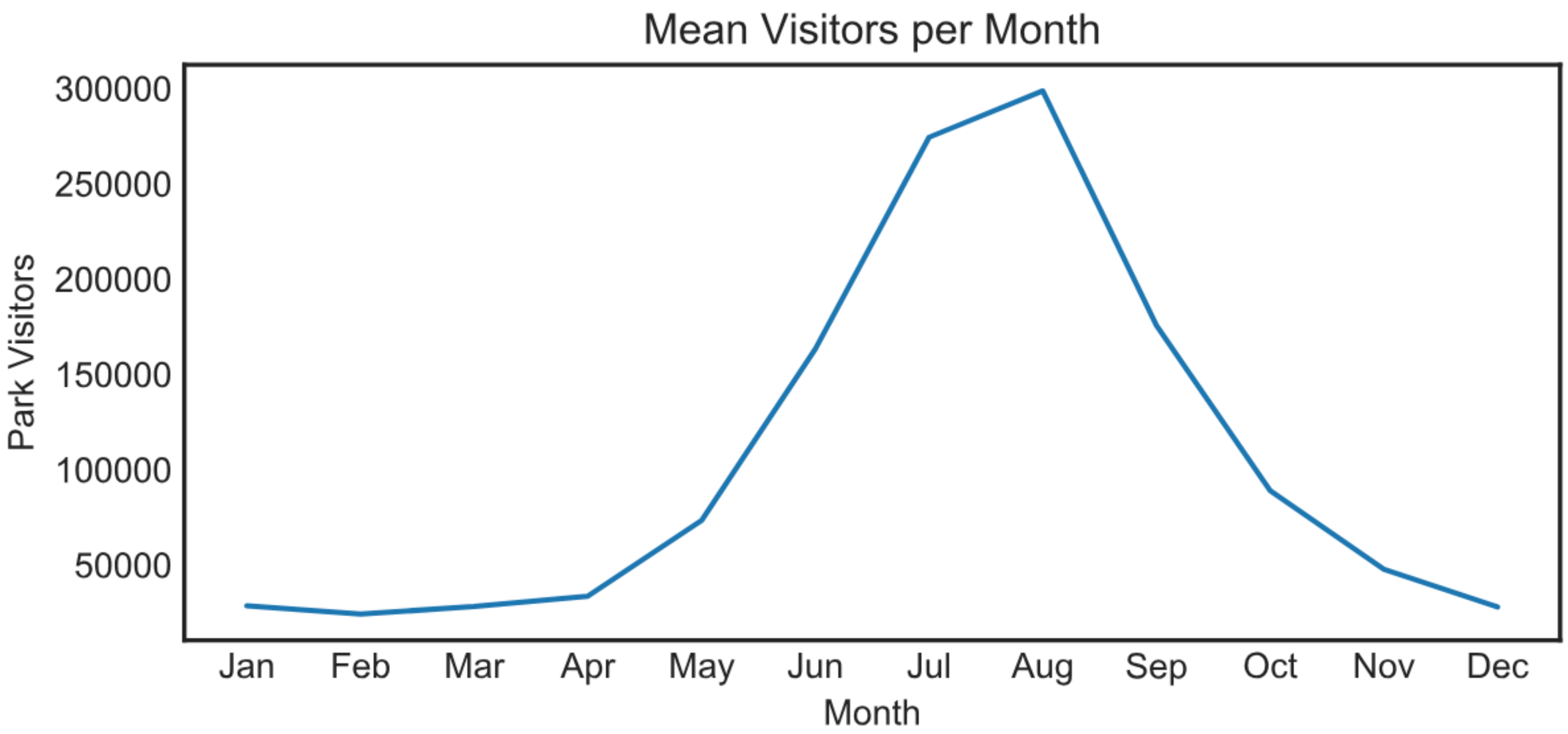
- More on data exploration
- More on transformation choices
- Normalized predictions vs actuals
- Calculating percentage and actual error, MSE

Data Exploration



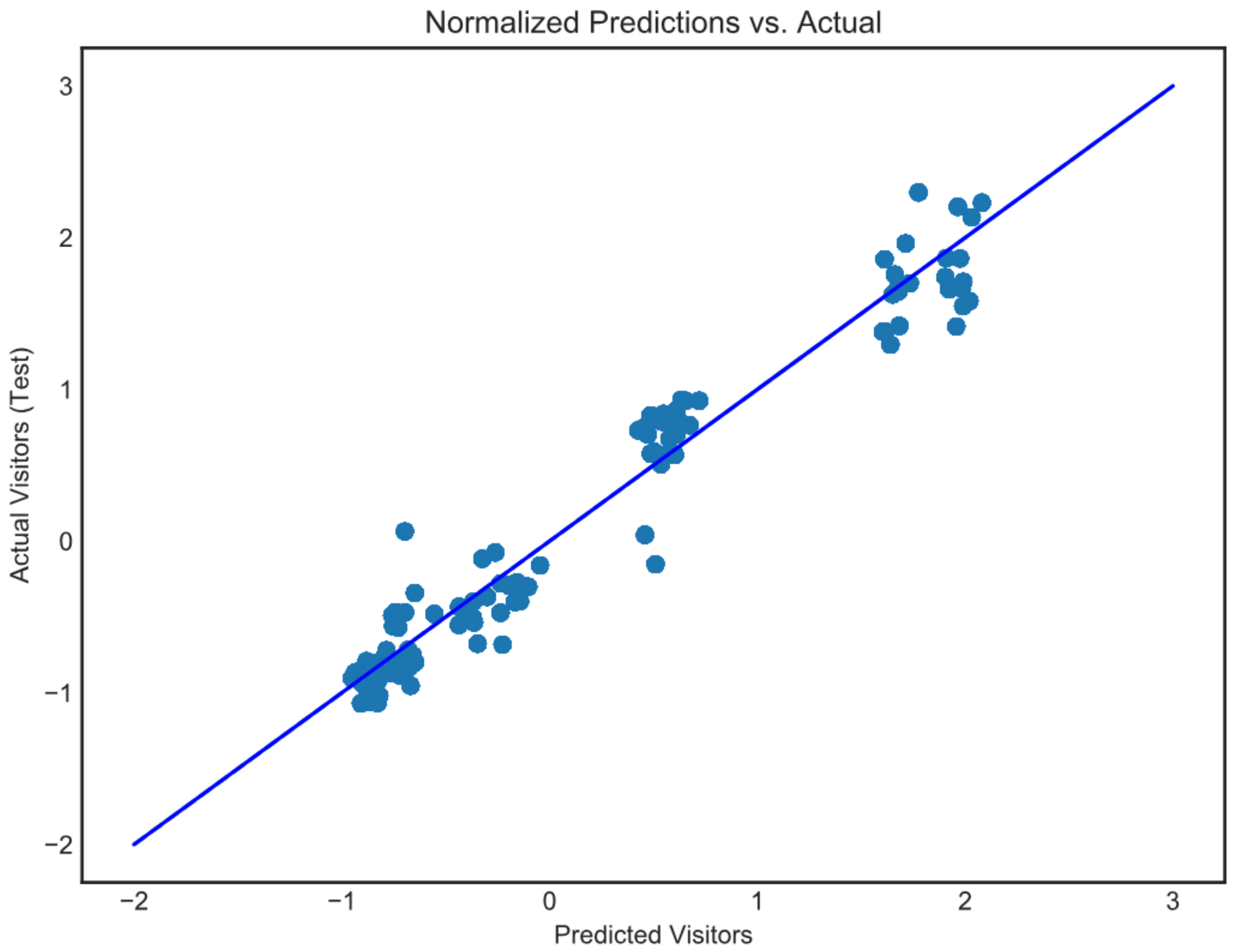
Data Exploration Continued

- Months might need transformation:



- Linear (Months 1-12) – but non-linear relationship: Adj R^2 of .857
- Polynomial to fit: Adj R^2 of .866
- Dummies to capture month-by-month effect: Adj R^2 of .952

Normalized Validations



Calculating percentage + actual errors, MSE

```
In [75]: np.mean(abs(y_pred2 - y_test2)/y_test2)

Out[75]: VISITORS    0.769318
         dtype: float64

In [83]: np.median(abs(y_pred2 - y_test2)/y_test2)

Out[83]: 0.15529121892923237

In [76]: np.mean(abs(y_test2 - y_pred2))

Out[76]: VISITORS    14030.788567
         dtype: float64

In [87]: np.median(abs(y_test2 - y_pred2))

Out[87]: 9385.969390615282
```

```
train_error = mean_squared_error(y_train2, lr.predict(X_train2))
test_error = mean_squared_error(y_test2, y_pred2)
mean_error = mean_squared_error(y_train2, y_mean2)
print(train_error)
print(test_error)
print(mean_error)
```

```
513532607.12683654
373874271.64138234
9687755710.36165
```

Model Validations

- High Adjusted R^2
- Cross Validation
  - Time series data, split data before and after a certain year
  - Across multiple CVs, Adj R^2 for predictions ranged from .95 to .97
  - Percentage difference in predicted and actual visitors (non-normalized)
    - Mean is 77%, might have outliers
    - Median is 15%
- High MSE (hundreds of millions), but we're working with large numbers
  - Just guessing the mean produces an MSE in tens of billions
  - Train error actually larger than test error, definitely not overfitting